DISSERTAÇÃO DE MESTRADO

MODELAGEM ANALÍTICA DA VAZÃO DE REDES SEM FIO BASEADAS NA NORMA IEEE 802.11ah

Stephanie Miranda Soares

Brasília, julho de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO

MODELAGEM ANALÍTICA DA VAZÃO DE REDES SEM FIO BASEADAS NA NORMA IEEE 802.11ah

Stephanie Miranda Soares

Dissertação de Mestrado submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia de Sistemas Eletrônicos e Automação

Banca Examinadora

Prof. Marcelo Menezes de Carvalho, Ph.D, FT/UnB _____
Orientador

Prof. Renato Mariz de Moraes, Ph.D, CIn/UFPE _____
Examinador interno

Prof. Priscila América Solis M. Barreto, Ph.D, CIC/UnB _____
Examinador externo

FICHA CATALOGRÁFICA

SOARES, STEPHANIE
MODELAGEM ANALÍTICA DA VAZÃO DE REDES SEM FIO BASEADAS NA NORMA IEEE 802.11ah [Distrito Federal] 2018.
xvi, 78 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2018).
Dissertação de Mestrado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. IEEE 802.11ah	2. Janela de acesso restrito (RAW)
3. Modelagem analítica	4. Probabilidade de término do slot RAW
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SOARES, S.M. (2018). *MODELAGEM ANALÍTICA DA VAZÃO DE REDES SEM FIO BASEADAS NA NORMA IEEE 802.11ah.* Dissertação de Mestrado, Publicação PGEA.DM-699/2018, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 78 p.

CESSÃO DE DIREITOS

AUTOR: Stephanie Miranda Soares
TÍTULO: MODELAGEM ANALÍTICA DA VAZÃO DE REDES SEM FIO BASEADAS NA NORMA IEEE 802.11ah.
GRAU: Mestre em Engenharia de Sistemas Eletrônicos e Automação ANO: 2018

Stephanie Miranda Soares
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

Dedico este Mestrado à minha mãe pelo incentivo e apoio em todas as minhas escolhas e decisões

Stephanie Miranda Soares

Agradecimentos

Agradeço a Deus por ter me dado saúde e força para chegar até aqui. Ao meu orientador Marcelo Menezes de Carvalho pelos seus ensinamentos, incentivo, paciência e por todo esforço para que eu concluísse o mestrado. A toda minha família, especialmente à minha mãe, por todo amor, apoio e incentivo. Ao Thiago pelo apoio e companheirismo. Por fim, agradeço a todos os professores e colegas que fizeram parte desta jornada.

Stephanie Miranda Soares

RESUMO

Inúmeras tecnologias promissoras foram desenvolvidas desde a concepção da Internet das Coisas (IoT, do inglês *Internet of Things*) que poderão modificar diferentes aspectos em nosso cotidiano. Tendo em vista o rápido crescimento da IoT, foi criado o grupo de tarefa IEEE 802.11ah para o desenvolvimento de uma nova norma Wi-Fi para lidar com os principais desafios da IoT, que são a conectividade de muitos dispositivos com recursos de energia limitados a um único ponto de acesso e o aumento do alcance das transmissões para distâncias muito maiores do que aquelas tipicamente utilizadas em redes locais sem fio. Assim, esse padrão traz novos mecanismos nas camadas física e de enlace para atender aos requisitos da IoT. Na camada física, houve modificações para operação em faixas de frequências abaixo de 1 GHz e, na camada MAC, o padrão traz novos mecanismos de economia de energia para ter um melhor desempenho em redes densas, sendo um deles o acesso restrito ao canal (RAW, do inglês *Restricted Access Window*). Recentemente, há muitos trabalhos sobre o IEEE 802.11ah, entretanto, a maior parte deles trazem ideias de agrupamento ou modelagem analítica para descobrirem o tamanho ótimo do RAW para obter maior desempenho da rede. No entanto, para um melhor entendimento dos efeitos dos diversos parâmetros e mecanismos do IEEE 802.11ah, esta dissertação apresenta um modelo analítico da função de coordenação distribuída do IEEE 802.11ah para o caso em que as estações estão distribuídas uniformemente em diversos slots RAW e possuem a mesma prioridade de tráfego, sob condições de tráfego saturado e canal ideal. Em nosso modelo, consideramos que cada intervalo de *beacon* possui somente um RAW o qual é dividido em slots RAW, e estudamos o impacto no desempenho da vazão ao variar o número de slots RAW e o número de estações na rede. Além disto, introduzimos ao modelo Markoviano uma probabilidade de término do slot RAW e propomos duas expressões empíricas para o cálculo desta probabilidade. Por fim validamos o modelo analítico com simulações no simulador de redes NS3 e, a partir das simulações computacionais, encontramos uma constante de ajuste para aproximar o modelo das simulações.

ABSTRACT

Many promising technologies have been developed since the conception of the Internet of Things (IoT) that can modify different aspects of our daily lives. In light of the rapid growth of IoT, it was created the IEEE 802.11ah Task Group to develop a new Wi-Fi standard to address IoT's key challenges, which are the connectivity of many devices with limited power resources to a single access point and the increase in the transmissions range for distances much greater than those typically used in wireless local area networks. Thus, this standard brings new mechanisms in the physical and link layers to meet IoT requirements. In the physical layer, there were modifications to permit the operation in frequency bands below 1 GHz and, in the MAC layer, the standard brings new power saving mechanisms to perform better in dense networks, one of which is the restricted access to the channel (RAW). Recently, there have been a lot of work on IEEE 802.11ah, however, most of them bring grouping or analytical modeling ideas to find out the optimal RAW size for a higher network performance. On the other hand, for a better understanding of the effects of the various IEEE 802.11ah parameters and mechanisms, this dissertation presents an analytical model of the distributed coordination function of IEEE 802.11ah for the case where the stations are evenly distributed in several RAW slots and have the same traffic priority, under conditions of saturated traffic and ideal channel. In our model, we consider that each beacon interval has only one RAW which is divided into RAW slots, and we study the impact on throughput performance by varying the number of RAW slots and the number of stations in the network. In addition, we introduce to the Markovian model a probability of reaching the end of a RAW slot and propose two heuristic expressions for this probability calculation. Finally we validate the analytical model with simulations in the networks simulator NS3 and, from the computational simulations, we find an adjustment constant to approximate the model of the simulations.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

q	Probabilidade de término do slot RAW
p	Probabilidade de colisão de um quadro de dados
P_s	Probabilidade de sucesso de um quadro de dados
P_{tr}	Probabilidade haver pelo menos uma transmissão no canal

Símbolos Gregos

δ	Atraso de propagação
σ	Tempo em que o slot está vazio
τ	Probabilidade da estação transmitir um pacote

Siglas

AP	Access Point
CSB	Cross Slot Boundary
DCF	Distributed Coordination Function
EDCA	Enhanced Distributed Channel Access
IFS	Interframe Space
EIFS	Extended Interframe Space
SIFS	Short Interframe Space
DIFS	DCF Interframe Space
PIFS	PCF Interframe Space
AIFS	Arbitration Interframe Space
IoT	Internet of Things
MIMO	Multiple Input Multiple Output
MTC	Machine Type Communication
M2M	Machine-to-Machine
RAW	Restricted Access Window
RPS	RAW Parameter Set
TIM	Traffic Indication Map
TWT	Target Wake Time

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A Internet das Coisas consiste em milhares de dispositivos conectados à Internet, e é uma revolução tecnológica que promete modificar nosso estilo de vida, permitindo que as "coisas" possam sentir o ambiente a sua volta e compartilhar essas informações, não só entre si, mas também com os seres humanos, para uma tomada de decisão inteligente em qualquer situação. A Internet das Coisas afetará positivamente o nosso cotidiano, pois dentre suas diversas aplicações pode-se citar automação de casas e indústrias, sistemas inteligentes para agricultura, como controle de irrigação e temperatura, monitoramento de pacientes em hospitais e sistemas inteligentes de transporte [2]. Vários estudos independentes concluíram que haverá um grande crescimento de indústrias IoT (do inglês *Internet of Things*) e M2M (do inglês *Machine-to-Machine*) nos próximos dez anos e, segundo um estudo realizado pela Ericsson em 2016 [3], a quantidade de dispositivos M2M conectados ultrapassará a quantidade de telefones celulares, computadores pessoais, laptops e tablets até 2020.

As redes de comunicação sem fio locais, WLANs (do inglês *Wireless Local Area Networks*), terão uma atuação importante na implementação e disseminação da IoT. No entanto, os padrões Wi-Fi atuais com protocolos de camada MAC (do inglês *Medium Access Control*) baseados em contenção, como é o caso da norma IEEE 802.11ac, encontrarão grandes desafios em cenários IoT, que consiste em redes com muitos dispositivos conectados com recursos de energia limitados e implantados em áreas muito amplas. Além disso, em redes de larga escala, é comum haver dezenas de pontos de acesso próximos uns aos outros [4], o que também ocasionará problemas de contenção em futuras redes IEEE 802.11. Deste modo, o IEEE 802.11ah *Task Group* (TGah) tem trabalhando em um novo padrão Wi-Fi que atenda aos principais desafios da IoT, que são a conexão de grande quantidade de dispositivos com recursos de energia limitados e a transmissão a longas distâncias [5]. O IEEE 802.11ah vem com modificações na camada física para operar nas faixas de frequência abaixo de 1 GHz, o que permite alcançar maiores distâncias em comparação com IEEE 802.11ac. Novos recursos foram introduzidos para que o IEEE 802.11ah alcance distâncias de até 1 km com uma taxa de dados de 100 Kb/s [6].

Na camada MAC, o IEEE 802.11ah trouxe mecanismos de economia de energia, sendo a janela de acesso restrito ao canal (RAW, do inglês *Restricted Access Window*) uma das principais inovações. A proposta do mecanismo RAW é dividir as estações da rede em grupos menores para que somente um grupo de cada vez possa disputar pelo canal em um determinado intervalo de tempo enquanto as estações que pertencem a outros grupos ficam no estado inativo. No IEEE 802.11ah, o tempo é dividido em intervalos de *beacon*, nos quais pode haver um ou mais RAWs, e cada RAW pode ser dividido em um ou mais slots. Deste modo, as estações podem tentar o acesso ao canal somente no seu slot RAW atribuído, com isso, resultando em uma diminuição da

disputa pelo acesso ao canal. Consequentemente, a probabilidade de colisão diminui durante o acesso ao canal, e as estações passam menos tempo no estado ativo, evitando o gasto de energia desnecessário. Esta proposta de agrupamento já foi amplamente utilizada a partir da ideia de agrupamento "*clustering*" em redes de sensores e redes sem fio *ad hoc*, em que as estações são geralmente agrupadas de acordo com a sua localização geográfica e um "*cluster-head*" é selecionado para coordenar as atividades de cada *cluster* [7], [8] e [9]. Todavia, o agrupamento por *clustering* necessita do envio de mensagens constantemente para atualizar localização das estações e para manter a hierarquia dos *clusters-head*, o que consumiria mais largura de banda e mais energia das estações, sendo que estas características não são desejáveis em redes no contexto de Internet das Coisas.

Recentemente, muitos trabalhos de pesquisa sobre o IEEE 802.11ah têm sido feitos, sendo que a maior parte deles trazem novas propostas de agrupamento das estações em grupos RAW a fim de otimizar o desempenho da rede, pois os critérios escolhidos para distribuir as estações em grupos RAW têm influência significativa no desempenho rede. Entretanto, ainda existem poucos trabalhos que desenvolveram modelos analíticos para analisar o IEEE 802.11ah e, entre eles, muitos fizeram um modelo matemático com o intuito de descobrir a duração ótima do slot RAW para melhorar o desempenho da rede. Dada a importância de modelos analíticos para a análise de desempenho de um protocolo de comunicação, e o fato de que na literatura há muitos trabalhos com propostas de modelos com cadeias de Markov que representem a dinâmica de redes baseadas no IEEE 802.11, neste trabalho, trouxemos a ideia de modelar o comportamento de uma estação baseada no IEEE 802.11ah a partir do seu processo de recuo segundo a função de coordenação distribuída (DCF, do inglês *Distributed Coordination Function*), baseado em uma cadeia de Markov. Sendo assim, esta dissertação apresenta um modelo analítico baseada em uma cadeia de Markov discreta para o comportamento do processo de *backoff* de uma estação que deseja transmitir seu pacote para analisar o desempenho da vazão de uma rede que opera sob o padrão IEEE 802.11ah com o tráfego saturado. Em nosso modelo, consideramos uma rede com um único grupo RAW em cada intervalo de *beacon*, sendo este grupo dividido em slots RAW nos quais as estações foram atribuídas uniformemente entre os slots RAW.

Uma particularidade do modelo apresentado é a tentativa de modelagem da probabilidade de finalização do slot RAW durante a atividade da estação, quando esta interrompe sua atividade para esperar pelo próximo slot RAW no intervalo de beacon subsequente. Nesta dissertação, apresentamos duas propostas empíricas para modelagem desta probabilidade, e analisamos o desempenho do modelo frente a simulações computacionais realizadas com o NS3. Além disso, estudamos o impacto desta probabilidade na fidelidade do nosso modelo, apresentando um estudo sobre o valor numérico necessário para uma constante que, aplicada à esta probabilidade, resulta em uma aproximação mais fidedigna do modelo aos resultados simulados. Este estudo servirá para futuras propostas empíricas para a probabilidade de finalização de slot RAW, crítica para este problema. Por fim, modelamos a vazão de um dado slot RAW e, em seguida, calculamos a vazão agregada do sistema, ou seja, a vazão resultante de todos os slots dentro do RAW, e validamos o modelo com simulações realizadas no simulador de redes NS3 utilizando o módulo

IEEE 802.11ah implementado por Le Tian et al. [1].

1.2 OBJETIVOS

Este trabalho tem como objetivo modelar o comportamento de uma estação baseada na norma IEEE 802.11ah a partir do seu processo de *backoff*, segundo a função de coordenação distribuída (DCF) em uma rede de tráfego saturado e canal ideal, ou seja, sem efeitos de propagação de sinal no canal. Utilizamos uma cadeia de Markov discreta bi-dimensional para modelar o processo de recuo (*backoff*) do DCF em uma estação que deseja transmitir um pacote dentro do seu slot RAW, em que os estados da cadeia representam o contador do recuo exponencial binário e as linhas são os estágios de retransmissão. Consideramos que cada intervalo de *beacon* possui somente um RAW, o qual é dividido entre slots RAW e as estações são distribuídas uniformemente entre eles, a fim de analisar o impacto no desempenho da vazão da rede ao dividir o acesso ao canal das estações em slots. Este tipo de modelagem com cadeias de Markov é muito comum em estudos sobre IEEE 802.11 DCF, todavia, o principal desafio em utilizar este tipo de abordagem para o IEEE 802.11ah é introduzir ao modelo os períodos em que a estação permanece no estado inativo, uma vez que a cadeia de Markov modela o comportamento dinâmico da rede. Desta forma, este trabalho também tem como objetivo incorporar ao modelo Markoviano uma probabilidade de "término do slot" e um cálculo da vazão que introduza os períodos em que a estação fica inativa. Por fim, validaremos o modelo com simulações no simulador de redes NS3. Este trabalho tem inspiração no trabalho [10], onde foi utilizada uma modelagem parecida para o IEEE 802.11 DCF no modo de economia de energia.

1.3 CONTRIBUIÇÕES

As contribuições desta dissertação podem ser assim resumidas:

- Proposta de um modelo analítico baseado em [10], [11] e [12] para a função de coordenação distribuída do IEEE 802.11ah para o caso de tráfego saturado e canal ideal (ou seja, sem efeitos de propagação de sinais) e quando há um único grupo RAW dividido em diferentes slots, aos quais as estações da rede foram distribuídas uniformemente .
- Proposta de expressões empíricas para o cálculo da probabilidade de término do slot RAW.
- Validação do modelo analítico via solução numérica confrontada com simulações realizadas no simulador computacional a eventos discretos NS-3.
- Estudo da precisão do modelo analítico desenvolvido para o IEEE 802.11ah frente às diferentes escolhas para o modelo empírico adotado para a probabilidade de término do slot RAW.

1.4 APRESENTAÇÃO DO MANUSCRITO

O restante deste trabalho está divido da seguinte forma. No Capítulo 2 é apresentada a fundamentação teórica necessária para o entendimento do trabalho, focando nas principais características do IEEE 802.11ah utilizadas no modelo teórico, e também é feita uma revisão bibliográfica de trabalhos anteriores relacionados ao IEEE 802.11ah. Em seguida, o Capítulo 3 apresenta o modelo analítico para operação do IEEE 802.11ah, sob tráfego saturado via cadeia de Markov discreta, assim como a sua solução, e as equações necessárias para o cálculo da vazão agregada. O cenário considerado no modelo e nas simulações e os resultados obtidos com o modelo e as simulações no NS3 são apresentados no Capítulo 4. Por fim, o Capítulo 5 traz as conclusões deste trabalho e propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTRODUÇÃO

A Internet das Coisas consiste em milhares de dispositivos com recursos de energia e potência limitados conectados a Internet. A fim de lidar com um dos maiores desafios da Internet das Coisas, é necessário um protocolo de comunicação que abranja cenários de redes densas, cujos dispositivos estão dispostos em grandes áreas, ou seja, o protocolo deve garantir a conectividade entre muitos dispositivos, alcançar longas distâncias e ter um baixo consumo de energia. Com o objetivo de resolver esse desafio, o IEEE 802.11ah Task Group desenvolveu uma nova versão do Wi-Fi para comunicação nas faixas de frequência abaixo de 1GHz, o IEEE 802.11ah, comercialmente conhecido como Wi-Fi HaLow. O novo padrão IEEE 802.11ah de comunicação sem fio que define as camadas Física e de Enlace, cujas características são uma combinação de características do padrão IEEE 802.11, o Wi-Fi, com tecnologias de comunicação de baixa potência, como o ZigBee e o Bluetooth. Foi desenvolvido para obter comunicação com alcance acima de 1Km, enquanto mantém a vazão de dados por volta de 150Kbps, oferencendo maior cobertura de área com uma vazão consideravelmente maior que as vazões obtidas em tecnologias de baixa potência. Na camada MAC, o IEEE 802.11ah introduz mecanismos como cabeçalho reduzido, associação rápida, janela de acesso ao canal restrito e agendamento de transmissão. Neste capítulo serão apresentados as principais características do IEEE 802.11ah utilizadas na realização deste trabalho.

2.2 O PADRÃO IEEE 802.11AH

Pensando nos cenários futuros da comunicação sem fio, o IEEE 802.11ah foi desenvolvido para suportar transmissões de longas distâncias, com mais de 8000 nós conectados a um único ponto de acesso e possui alta eficiência no consumo de energia, o que o torna um padrão de comunicação atraente para as aplicações de Internet das Coisas, como monitoramento por meio de redes de sensores e automação residencial. Em uma comparação com o protocolo IEEE 802.15.4 [13], o atual protocolo para dispositivos de baixa potência e com restrições quanto ao consumo de energia e memória, o IEEE 802.11ah possui um desempenho melhor em cenários com muitos dispositivos conectados. O IEEE 802.11ah herdou a camada física do IEEE 802.11ac, com adaptações para operar em frequências abaixo de 1GHz (que difere de acordo com regulamento de cada país $863 - 868$MHz na Europa e $902 - 928$MHz na América do Norte, por exemplo)[2]. A largura de banda vai de 1MHz a 16MHz, sendo que as bandas de 1MHz e 2MHz foram amplamente adotadas. Ao operar em frequências mais baixas e uma banda mais estreita, o IEEE 802.11ah alcança maiores distâncias com o consumo de energia menor que o Wi-Fi tradicional,

que utiliza frenquência de $2,4$GHz e largura banda de 5MHz. O 802.11ah herdou 10 configurações de esquemas de modulação e codificação (MCSs, do inglês *Modulation and Coding Schemes*) com diferentes taxas de entrega de dados e confiabilidade, conforme a Tabela 2.1 [1]. Além disso, assim como o 802.11ac, o 802.11ah utiliza OFDM (do inglês Orthogonal Frequency Division Multiplexing), MIMO (do inglês Multiple Input Multiple Output) e DL MU-MIMO (do inglês Dowklink Multi-User MIMO) [14].

Tabela 2.1: Tabela de esquemas de modulação e codificação (MCSs) para 1MHz e 2MHz com intervalo de guarda $GI = 8\ \mu$s

Índice MCS	**Modulação**	**Taxa de Codificação**	**Taxa de Dados (Kbps) 1 MHz**	**Taxa de Dados (Kbps) 2 MHz**
0	BPSK	1/2	300	650
1	QPSK	1/2	600	1300
2	QPSK	3/4	900	1950
3	16-QAM	1/2	1200	2600
4	16-QAM	2/3	1800	3900
5	64-QAM	1/2	2400	5200
6	64-QAM	3/4	2700	5850
7	64-QAM	5/6	3000	6500
8	256-QAM	3/4	3600	7800
9	256-QAM	5/6	4000	Inválido
10	BPSK	1/2 com 2x repetições	150	Inválido

Na camada MAC, foram introduzidas várias características com o intuito de atender às condições da IoT, entre elas, cabeçalho reduzido, associação e autenticação mais rápidas, janela de acesso restrito ou *restricted access window (RAW)*, *traffic indication map (TIM)* e *target wake time (TWT)* [15]. Para associação e autenticação mais eficientes, o 802.11ah centraliza esse processo, em que o ponto de acesso configura um *threshold* no *beacon*. A estação inicializa e gera um valor aleatório no intervalo de $[0, 1022]$ e envia o pedido de associação/autenticação ao ponto de acesso. Se esse valor aleatório for menor que o *threshold* indicado no *beacon* recebido, a estação deve adiar a associação/autenticação para o próximo *beacon*. O ponto de acesso sempre atualiza esse *threshold* para que todas as estações possam associar-se ao ponto de acesso eventualmente.

Quanto o acesso ao canal, a fim de evitar colisão e obter uma vazão mais alta em redes densas, o IEEE 802.11ah introduz a janela de acesso restrito, RAW, a qual divide as estações em grupos RAW. O tempo é dividido em intervalos, em que cada um é atribuído para um grupo RAW, e somente as estações que pertencem ao grupo RAW podem acessar o canal no intervalo determinado. Mais detalhes sobre o funcionamento do RAW serão dados na seção 2.3. Há ainda o mecanismo de segmentação TIM, o qual divide a informação em vários segmentos para transmti-los separadamente e, junto ao TIM *beacon*, tem o *Delivery Traffic Indication Map (DTIM) beacon*. O ponto de acesso envia o DTIM *beacon* em broadcast para as estações as quais possuem segmentos TIM com dados pendentes, deste modo, as estações acordam apenas para ouvir seu TIM *beacon* correspondente, permanecendo por mais tempo no modo de economia de energia. A redução do

consumo de energia pode ser ainda maior para estações que raramente possuem pacotes para enviar utilizando o TWT. Com o TWT, as estações podem negociar com o ponto de acesso quando enviarão seus pacotes, mantendo-se em economia de energia por mais tempo.

Nas Seções 2.3 e 2.4 serão dados mais detalhes das características do IEEE 802.11ah relevantes para o modelo matemático desenvolvido neste trabalho e na Seção 2.5 será explicado o módulo do IEEE 802.11ah para o simulador de redes NS-3 criado por Le Tian et al. [16].

2.3 JANELA DE ACESSO RESTRITO (RAW)

Com o intuito de reduzir o número de colisões quando muitas estações disputam pelo canal simultaneamente, no acesso restrito ao canal, o RAW, as estações são dividas em grupos RAW e somente as estações que pertencem ao grupo podem acessar o canal no intervalo de tempo determinado. Assim como no modo de economia no IEEE 802.11 tradicional, o tempo é dividido em intervalos de beacon e cada intervalo de beacon é precedido por um beacon que carrega as informações do RAW, o RPS (do inglês *RAW Parameter Set*), o qual especifica quais estações pertencem ao RAW, a duração e o tempo de início. O mecanismo RAW é representado na Fig. 2.1.

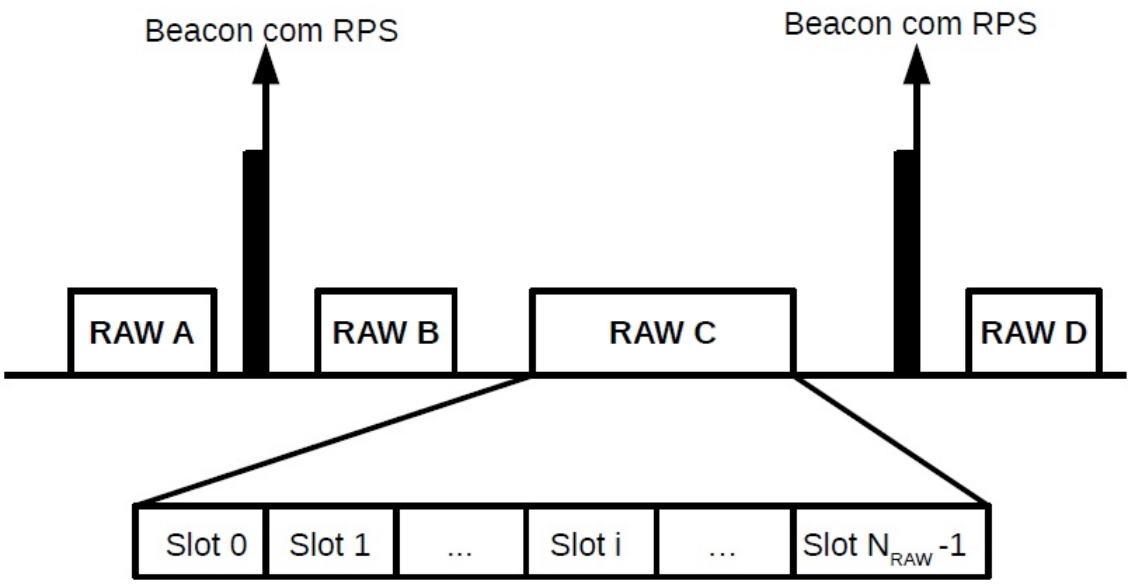

Figura 2.1: Funcionamento do mecanismo RAW no IEEE 802.11ah [1].

Além disso, cada intervalo de beacon pode ter mais de um RAW com parâmetros diferentes. O RAW também tem seu tempo dividido em slots, os quais serão atribuídos para as estações tentarem o acesso ao canal, e cada RAW pode ter um ou mais slots, assim, o RPS também contém o número de slots em cada RAW e a duração dos slots. A duração D de cada slot, em μs é determinada pela Eq.(2.1):

$$D = 500 + C \times 120, \tag{2.1}$$

em que C representa o subcampo de contagem de duração do slot no RPS e os valores de D e C são determinados a partir da duração total do RAW e da quantidade de slots contidos no RAW.

Diferente dos outros padrões IEEE 802.11, cada estação utiliza dois contadores de *backoff* do EDCA (do inglês *Enhanced Distributed Channel Access*), o primeiro é utilizado fora do RAW atribuído e o segundo dentro do RAW. A estação executa o primeiro *backoff* fora do RAW, suspendendo-o no início de cada RAW e armazena o estado o qual parou para retornar ao final do RAW. Se a estação faz parte do RAW, a estação inicia o segundo *backoff* dentro do seu respectivo slot RAW e descarta o estado de *backoff* em que parou ao terminar o slot. Na Fig. 2.2, a estação 1 pertence ao grupo RAW e foi atribuída ao slot 1, enquanto a estação 2 não faz parte deste grupo RAW. Portanto, a estação 1 realiza o primeiro *backoff* fora do RAW e o segundo *backoff* dentro do seu slot RAW, e a estação 2 realiza o primeiro *backoff* fora do RAW e vai para o estado de dormência durante o RAW. As duas estações retornam ao primeiro *backoff* ao término do RAW [1].

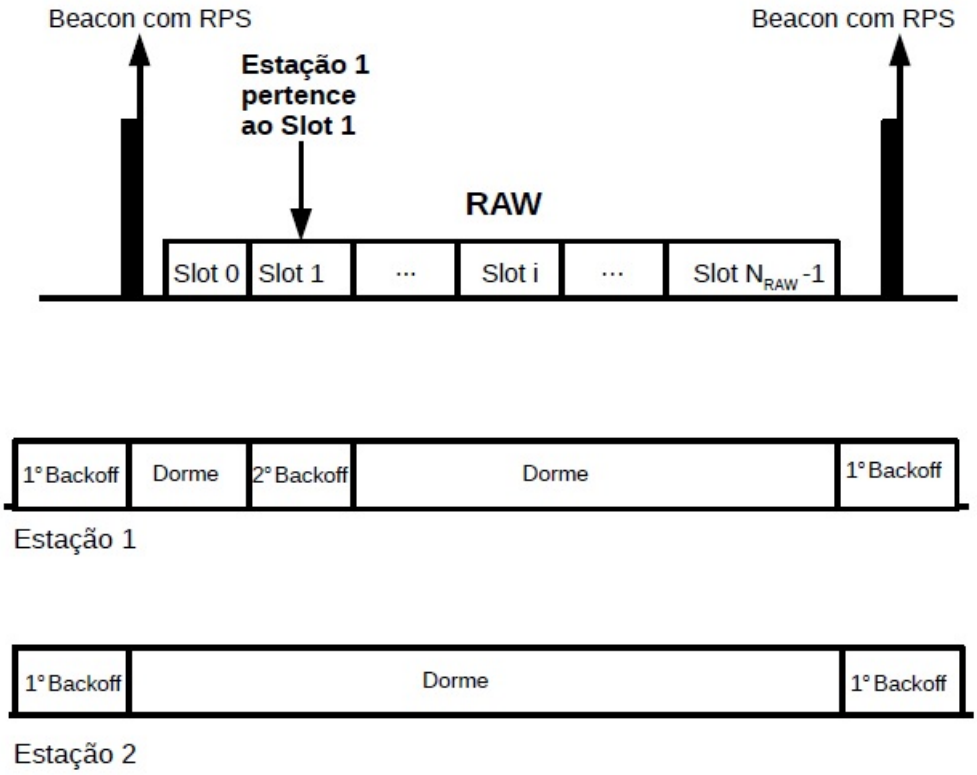

Figura 2.2: Procedimento do *backoff* dentro do RAW [1].

De acordo com o padrão [17], uma transação que está ocorrendo pode ou não ultrapassar o slot RAW. No caso em que é permitido a transmissão ultrapassar o slot, o CSB (do inglês *Cross Slot Boundary*) é habilitado, caso contrário o CSB é desabilitado. Quando o CSB está desabilitado, a

estação só iniciará sua transmissão se houver tempo hábil para transmitir seu pacote e receber um ACK, o qual denominamos tempo de holding T_h. Além do tempo de holding, há ainda o tempo de guarda para proteger o slot RAW e evitar que a transmissão ultrapasse seu término, caso haja erros de sincronização ou atraso de propagação maior que o esperado.

2.4 *ENHANCED DISTRIBUTED CHANNEL ACCESS (EDCA)*

A camada MAC do IEEE 802.11ah provê os serviços do EDCA por meio dos serviços do DCF (do inglês *Distributed Coordination Function*). Como uma extensão do DCF [18], no EDCA, a estação com maior prioridade de tráfego tem mais chances de transmitir do que uma estação com prioridade de tráfego menor. O EDCA suporta até oito prioridades em uma estação, as quais são mapeadas em quatro categorias de acesso diferentes, assim, as janelas de contenção mínima e máxima dependem de cada categoria de acesso e a estação deve aguardar por um tempo AIFS, menor que o DIFS do DCF tradicional, para tentar transmitir o pacote quando o canal está ocioso o qual também depende da categoria de acesso. No modelo e nas simulações, as estações possuem a mesma prioridade de tráfego, portanto, as estações operam sob o DCF tradicional do IEEE 802.11.

O DCF é baseado em um protocolo de acesso aleatório ao meio que evita colisão, o qual a estação, ao perceber que o canal está ocioso, poderá transmitir o seu pacote somente após o um intervalo de tempo denominado espaçamento interquadros (IFS, do inglês *Interframe Space*). O Padrão IEEE 802.11 classifica esses espaços entre quadros de acordo com a sua finalidade, como especificado a seguir.

- SIFS (*Short Interframe Space*): Espaço de tempo entre quadro utilizado para quadros de controle, que possuem maior prioridade, ou seja, um quadro de controle é transmitido após o término do SIFS;
- PIFS (PCF *Interframe Space*): Utilizado pela função de coordenação pontual, redes em que não há disputa pelo meio de comunicação. O coordenador define em qual slot de tempo a estação irá transmitir;
- DIFS (DCF *Interframe Space*): Tempo mínimo entre quadros em uma rede com disputa ao meio, na função de coordenação distribuída. Uma estação só transmitirá seu quadro se o canal estiver ocioso por um tempo maior ou igual ao DIFS;
- EIFS (*Extended Interframe Space*): Não possui um tamanho fixo e é utilizado quando ocorre erro na transmissão;
- AIFS (*Arbitration Interframe Space*): Utilizado pelas estações que possuem prioridade de tráfego no EDCA.

Portanto, uma estação que deseja transmitir um pacote verifica a atividade do canal, se o canal

estiver ocioso por um intervalo de tempo maior ou igual ao DIFS, a estação transmitirá sua mensagem e aguardará por um período igual a SIFS para receber a confirmação que sua mensagem chegou corretamente no receptor [19]. Caso o canal esteja ocupado, a estação espera um intervalo de *backoff* aleatório escolhido de acordo com o algoritmo de *backoff* exponencial enquanto perceber que o canal está ocioso novamente. Desta forma o protocolo minimiza a probabilidade de colisão com pacotes transmitidos por outras estações. A estação também aguarda o *backoff* aleatório entre a transmissão de dois pacotes consecutivos mesmo que o canal esteja ocioso por um período igual ou maior ao DIFS.

2.4.1 O Protocolo MAC do IEEE 802.11 - CSMA/CA

O CSMA-CA, protocolo MAC de acesso múltiplo por detecção de portadora com prevenção de colisão, permite que os pacotes de diversos dispositivos sejam transmitidos em um mesmo canal. Antes de iniciar uma transmissão, uma estação qualquer que tenha um quadro para transmitir verifica a atividade do canal. Se o canal estiver ocioso, a estação deve aguardar por um intervalo de tempo DIFS para iniciar sua transmissão. Entretanto, se o canal estiver ocupado, a estação escolherá um valor aleatório de *backoff* e fará a contagem regressiva durante o período em que perceber o canal ocioso. Caso ainda perceba que o canal ficou ocupado durante o *backoff*, o valor do contador permanecerá estacionário. Finalmente, quando o contador chegar a zero, a estação transmitirá o seu quadro inteiro. As tentativas de retransmissão ocorrem até que o canal esteja livre ou até que um limite de tentativas definidas pelo desenvolvedor da rede seja alcançado.

O Protocolo MAC do IEEE 802.11 ainda pode utilizar quadros curtos de controle para evitar colisão de pacotes de terminais ocultos como mencionado em [20]. Esses quadros de controle são o RTS (do inglês *Request to Send*), solicitação de envio, e o CTS (do inglês *Clear to Send*), pronto para envio. Antes de transmitir o quadro, a estação transmite um RTS em *broadcast* para reservar o canal e o ponto de acesso responde com um CTS informando que haverá uma transmissão neste canal. O quadro CTS é ouvido por todas as estações dentro da faixa de alcance do ponto de acesso, assim, uma estação transmite seu quadro de dados e todas as outras estações ficam sem transmitir mesmo que a estação transmissora não esteja dentro de suas faixas de alcance. O RTS/CTS ajuda a reduzir colisões entre pacotes melhorando o desempenho principalmente na transmissão de quadros longos, uma vez que, se houver colisão entre quadros RTS/CTS, a duração a qual o canal estará ocupado por uma colisão será menor. Porém o seu uso introduz um atraso e utiliza mais recursos da rede, podendo diminuir a vazão e consumir mais energia. Por este motivo, o RTS/CTS é normalmente mais utilizado para reservar o canal para a transmissão de quadros longos.

A Fig. 2.3 mostra um exemplo do funcionamento do CSMA/CA com o acesso básico ao meio, sem os quadros RTS/CTS. A estação *source* percebe que o canal está ocioso e aguarda um intervalo de tempo DIFS para enviar seu pacote de dados para a estação *destination*. Ao receber o pacote de dados com sucesso, a estação *destination* envia um ACK após um intervalo de tempo SIFS a fim de indicar o recebimento do pacote à estação *source*. As outras estações não transmi-

tem enquanto o canal estiver ocupado e, após perceber que o canal está ocioso novamente por um intervalo de tempo maior ou igual a DIFS, iniciam o processo de *backoff*, que será detalhado na Seção 2.4.2.

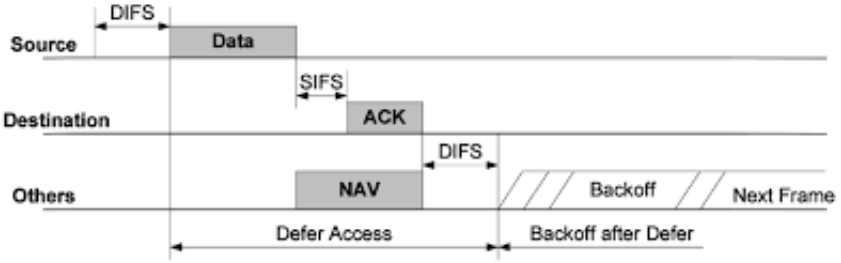

Figura 2.3: Acesso básico ao meio do CSMA/CA.

2.4.2 O Algoritmo de Recuo Exponencial Binário (BEB)

Como mencionado anteriormente, o DCF utiliza o algoritmo do Recuo Exponencial Binário (BEB, do inglês *Binary Exponencial Backoff*) para evitar colisão. Ao perceber que o canal está ocupado, a estação inicia uma contagem regressiva do intervalo de recuo a partir do momento em que percebe o canal ocioso novamente. A estação espera por um tempo DIFS a partir do momento em que o meio fica ocioso e em seguida define o temporizador do recuo [15]. O intervalo de recuo é escolhido de forma aleatória com a mesma probabilidade e é dado por:

$$\textit{Backoff time} = \textit{Random()} \times \textit{Slot time} \tag{2.2}$$

O valor de *Random()* está uniformemente distribuído no intervalo $[0, CW - 1]$, onde $CW_{min} \leq CW \leq CW_{max}$ e CW_{min} e CW_{max} são os tamanhos da janela de contenção mínima e da janela de contenção máxima, respectivamente. Quando o canal está ocioso, o temporizador do recuo é decrementado e, se o canal estiver ocupado, essa contagem congela até que o canal esteja ocioso novamente por um intervalo de tempo maior ou igual a DIFS. Inicialmente, a janela de contenção está definida para o valor mínimo e a cada transmissão sem sucesso a janela de contenção mínima aumenta exponencialmente, ou seja, $CW = 2^i CW_{min}$, onde i está no intervalo $[0, m]$ é o estágio de retransmissão, sendo m o estágio de retransmissão máximo e $CW = 2^m CW_{min}$.

2.5 O MÓDULO IEEE 802.11AH PARA O NS3

Le Tian et al. [1] adaptaram quatro principais componentes dos modelos da camada física e da camada de enlace do Wi-Fi no NS3 para criar um módulo que suporte o IEEE 802.11ah. Atualmente, o NS3 vem com suporte para alguns padrões IEEE 802.11, incluindo o 802.11a, 802.11b, 802.11g, 802.11n e o 802.11ac, que é o mais recente. A implementação é baseada na versão 3.23 do NS3 e os quatro principais componentes são:

- *WifiChannel*: Aproximação analítica do meio físico o qual o pacote é propagado, no caso do Wi-Fi é o ar, que consiste nos modelos de perda de propagação e de atraso.
- *WifiPhy*: É a parte da camada física do protocolo, envia e recebe quadros e determina a perda causada por interferência.
- *MacLow*: Implementa transmissões RTS/CTS/DADOS/ACK, DCF, EDCA, filas de pacotes, fragmentação, retransmissão e controle de taxa de dados.
- *MacHigh*: Implementa funções de gerenciamento, como geração de beacon e associação;

2.5.1 Alterações na Camada Física

Na camada física do IEEE 802.11ah, Le Tian et al. [1] implementaram os esquemas de modulação e codificação do MCS0 ao MCS9 para as bandas de canal 1, 2, 4, 8 e 16 MHz, conforme a Tabela 2.1, sendo que o MSC10 não foi implementado. Deste modo, o módulo do IEEE 802.11ah no NS3 permite utilizar diferentes taxas de dados, sendo $7,8$ Mb/s a taxa de dados máxima permitida. O cabeçalho e o preâmbulo do IEEE 802.11ah, assim como o cálculo da duração de envio e recebimento do preâmbulo, cabeçalho e dados foram implementados. Além disto, algumas funções forma modificadas para calcular o comprimento dos pacotes 802.11ah recebidos a fim de realizar o cálculo da taxa de erro de pacote.

Para as perdas de propagação de sinal, Le Tian et al. [1] implementaram um modelo que determina a força do sinal baseado na distância entre o remetente e o receptor. Foram implementados dois modelos de propagação para o IEEE 802.11ah, um para ambientes internos e o outro para ambientes externos, desenvolvidos por Hazmi et al. [21]. Portanto, Le Tian et al. [1] implementaram dois modelos de perdas de propagação para ambientes externos, os quais são utilizados em simulações de áreas extensas e de áreas densas, e um modelo de propagação para ambientes internos que é igual ao modelo de propagação do 802.11n. Além desses modelos de perdas de propagação, ainda é possível realizar as simulações para o caso de canal ideal, sem perdas de propagação, o qual foi utilizado neste trabalho.

2.5.2 Alterações na Camada MAC

O módulo do NS3 para o IEEE 802.11ah implementado por Le Tian et al. [1] suporta apenas o mecanismo de associação rápida e o mecanismo RAW. Os atributos relacionados ao RAW, como o número de grupos RAW, número de estações RAW, entre outros, foram adicionados para permitir que o usuário configure. Os elementos RPS e *AuthentCtrl*, que carregam as informações do RAW e de associação, respectivamente, são definidos em novas classes e incluídos no *beacon* com as configurações atualizadas a cada intervalo de *beacon*. Também foi implementado um mecanismo para ajustar o *threshold* de associação dinamicamente, em que estação recebe o RPS e o *AuthentCtrl* com o *threshold* de associação e, baseada nesse *threshold*, a estação decide se enviará o pedido de associação para o ponto de acesso. A estação também é capaz de decidir, a

partir das informações do RAW contidas no RPS, em qual slot RAW poderá acessar o canal.

Le Tian et al. [1] também implementaram os dois *backoffs* definidos na norma IEEE 802.11ah. A estação realiza o primeiro *backoff* e, ao iniciar um RAW, a estação para a contagem do primeiro *backoff* e armazena o valor o qual parou. Baseado nas informações contidas no elemento RPS, a estação inicia o segundo *backoff* e compete pelo canal no slot apropriado com o valor de estágio de *backoff* correspondente. Ao término do RAW, as estações retornam aos seus respectivos estágios do primeiro *backoff*. Algumas classes forma modificadas para armazenar e retornar ao estágio de retransmissão dos dois *backoffs*. Além disto, o módulo ainda suporta transmissões com qualidade de serviço (QoS) e sem QoS.

2.5.3 Trabalhos Relacionados

O crescimento de comunicações M2M e cenários IoT tornou-se significativo nos últimos anos, e seus desafios têm sido um assunto bastante estudado na literatura. Pelo fato de se tratar de cenários com redes densas cujos nós estão localizados em grandes áreas, há a necessidade de novas propostas de protocolos de comunicação para lidar com esses desafios, visto que os protocolos para redes sem fio existentes baseados em contenção não conseguiram suprir as necessidades de uma rede IoT. Por esse motivo, o TGah (do inglês *IEEE 802.11ah Task Group*) desenvolveu um novo padrão Wi-Fi, cujas principais modificações foram a transmissão em faixas de frequência abaixo de 1GHz e o agrupamento de estações (RAW) para alcançar distâncias maiores e diminuir a competição pelo canal, respectivamente. Por ser um padrão lançado recentemente, há uma grande discussão de como otimizar o mecanismo RAW, com propostas de técnicas de agrupamento ou modelagem da duração ótima do RAW a fim de otimizar o uso do canal. Contudo, ainda há poucos trabalhos com modelagens matemáticas para analisar o desempenho do IEEE 802.11ah como foi feito com o IEEE 802.11. Neste capítulo são apresentados os principais trabalhos relacionados ao IEEE 802.11ah.

Khorov et al. [22] fizeram uma aproximação matemática utilizando cadeia de Markov para analisar a eficiência do mecanismo RAW e adaptar seus parâmetros em cenários típicos de MTC. Os autores definiram dois processos: o processo A, em que uma estação aleatória transmite com sucesso seu pacote, e o processo B, em que todas as estações da rede transmitem seus pacotes com sucesso. O modelo consiste em duas cadeias de Markov para representar cada processo e modela o número de transmissões sem sucesso antes da primeira transmissão com sucesso de cada estação e as duas cadeias vão para estados absorventes. O caso A quando a estação transmite o pacote com sucesso ou atinge o limite máximo de retransmissão, e o caso B quando todas as estações transmitem seus pacotes com sucesso. A duração do slot RAW foi definida de tal forma que a transmissão de pacotes por uma estação escolhida ou por todas as estações termina ao final do slot com uma probabilidade predefinida, ou seja, os estados absorventes possuem probabilidades predefinidas pelos autores na solução das cadeias. As probabilidades dos estados e as probabilidades de transição foram obtidas computacionalmente, incrementando t, que representa o número de slots virtuais do slot RAW, até que as probabilidades dos estados absorventes excedessem a

probabilidade predefinida. Desta forma, foi possível encontrar a distribuição de tempo necessário para uma estação transmitir seu pacote com sucesso e a distribuição de tempo necessário para todas as estações da rede transmitirem seus pacotes com sucesso e, assim, determinar a duração mínima de um slot RAW. O modelo, assim como o proposto neste trabalho, também foi validado com o simulador NS3, entretanto, eles não utilizaram o módulo de [1].

Hazmi et al. [21] estudaram a viabilidade da tecnologia de rádio OFDM do IEEE 802.11ah em cenários IoT e M2M, observando perdas na propagação do sinal do transmissor para o receptor. Baseados nesses estudos, eles propuseram um modelo de perda de propagação, o qual foi utilizado por Le Tian et al. [1] na implementação do módulo IEEE 802.11ah no NS3. Raeesi et al. [23] propuseram um modelo analítico considerando o tráfego saturado e os dois tipos de acesso ao canal, básico e RTS/CTS, para computar a vazão e o consumo de energia e analisar o impacto ao utilizar o mecanismo RAW em redes densas, realizando uma comparação com o DCF convencional. Com uma abordagem diferente da utilizada neste trabalho, o modelo estima, por meio de análise do valor médio, o tempo médio de uma trasmissão com sucesso de um pacote considerando a quantidade de erros e de colisões antes da transmissão de sucesso, assumindo as probabilidades de erro e de colisão conhecidas. Ao obter o tempo médio de uma transmissão com sucesso, calcularam a vazão e o consumo de energia para o tráfego saturado e validaram o modelo com simulações no OMNET++. Adicionalmente, Raeesi et al. [23] estenderam suas simulações para um cenário mais realista, com múltiplos pontos de acessos e muitos dispositivos conectados, a fim de analisar os efeitos de sobreposição das áreas de alcance dos pontos de acesso e verificaram o desempenho da rede ao utilizar o mecanismo RAW. Concluíram, com seus resultados, a importância desse novo mecanismo para a melhora do desempenho do sistema em redes muito densas.

Park et al. [24] desenvolveram um algoritmo de aprimoramento da camada MAC do IEEE 802.11ah para estimar o número de dispositivos no acesso *uplink*. Baseado na quantidade de dispositivos que tentam acessar o canal, o algoritmo determina, por meio de estimação estocástica, o tamanho ótimo do RAW para que o canal seja utilizado de forma mais eficiente. Para esse estudo, assume-se que o intervalo de *beacon* possui um RAW de *uplink* e um RAW de *downlink*, em que a estação vencedora na disputa pelo acesso ao canal em um slot aleatório envia um PS-poll para o AP solicitando os dados de *downlink* ou um ACK em resposta ao pacote enviado, e o AP faz a estimação de quantos dispositivos realizaram o acesso *uplink* por meio do algoritmo. Desta forma o algoritmo considera dois casos para fazer a estimação, o caso em que há slots vazios no RAW e o caso em que não há slots vazios. Para o caso em que há slots vazios no RAW, o AP estima o número de dispositivos no acesso *uplink* pelo método de estimação por máxima verossimilhança, entretanto, como em redes com muitos dispositivos conectados provavelmente não haverá slots vazios no RAW, o AP estima o número de dispositivos com a probabilidade de sucesso dos acessos *uplink* a partir das probabilidades obtidas por Bianchi [25].

Ao introduzir o modelo de Bianchi no cálculo da probabilidade de sucesso, os autores consideraram que o comportamento da rede saturada no IEEE 802.11ah é semelhante ao comportamento do IEEE 802.11, pois as estações tentam acessar o canal por meio do processo de *backoff* e há

transmissões acontecendo constantemente dentro um slot RAW em condições de tráfego saturado da mesma forma de uma rede IEEE 802.11 saturada, contudo, o modelo não introduz o momento quando as estações vão para o estado inativo ao término do slot RAW. Além disso, o modelo markoviano de Bianchi representa um comportamento dinâmico da rede em regime estacionário, cujas estações estão sempre em estado ativo tentando transmitir pacotes, já no IEEE 802.11ah as estações estão sempre transitando entre os estados ativo e inativo, e a duração de um slot RAW pode não ser suficiente para a rede entrar em regime estacionário. Portanto, o modelo de Bianchi pode não representar adequadamente o comportamento de uma rede baseada no IEEE 802.11ah.

Um dos trabalhos mais completos em relação à modelagem analítica do IEEE 802.11ah foi o de Zheng et al. [26] que propuseram um modelo que permite estimar a vazão máxima em cenários de tráfego saturado, em que as estações são divididas igualmente em grupos e cada slot é atribuído a um grupo, considerando também que os slots não estão com o CSB (do inglês *Cross Slot Boundary*) habilitado. Para encontrar a probabilidade de colisão e a probabilidade de transmissão, foi utilizado o método do valor médio e, assim, estimaram a quantidade de transações em um período de acesso ao canal e a duração de cada transação. Zheng et al. [17] estenderam o modelo para o caso em que o CSB está habilitado. Nesse artigo, os autores assumem que a duração a qual o canal está disponível para contenção é menor do que a duração do slot RAW e que essa duração pode ser variável, pois deve-se considerar o tempo de *handover* entre os slots RAW. Portanto, Zheng et al. incrementaram o modelo com uma cadeia de Markov para estimar a quantidade de mini-slots ocupados na última transação e calcularam a duração do próximo slot RAW. Também foram realizadas simulações para avaliar e comparar o desempenho de redes densas sob dois tipos de agrupamento: o centralizado, em que o AP distribui uniformemente as estações entre grupos, e o descentralizado, em que a estação associa-se a um grupo aleatoriamente, e observaram que ambos agrupamentos podem ter desempenhos semelhantes quanto a vazão.

As ideias de agrupamento com o propósito de otimizar o desempenho do protocolo estão sendo amplamente exploradas nos trabalhos relacionados ao IEEE 802.11ah. A partir dos estudos realizados em [17], Hazmi et al. [27] propuseram um novo esquema de agrupamento baseado no estado de *backoff* das estações para melhorar a vazão e diminuir o consumo de energia do IEEE 802.11ah. Assumindo que o AP conhece os estados de *backoff* das estações, um algoritmo lista as estações em ordem crescente de *backoff* e as estações com *backoffs* iguais são colocadas em grupos diferentes. Com isto, obtém-se menos colisão e o aumento da vazão da rede. Chang et al. [28] argumentaram que o desempenho do mecanismo de agrupamento está relacionado às diferentes demandas de tráfegos dos dispositivos, por isso os grupos devem ser adaptados às demandas de tráfego. Deste modo, propuseram um esquema de agrupamento baseado no balanceamento de cargas e verificaram uma melhora quanto à eficiência no uso do canal e quanto à vazão.

Dong et al. [29] e Yoon et al. [30], desenvolveram esquemas de agrupamento de acordo com a localização geográfica dos dispositivos a fim de evitar colisões relacionadas ao problema do terminal escondido. Dong et al. [29] dividem a área de cobertura do AP em partes iguais e cada estação é atribuída ao slot RAW de acordo com a sua posição na área de cobertura, enquanto

Yoon et al. [30] propõem um algoritmo em que o AP detecta os terminais escondidos por meio da diferença de tempo entre as transmissões do PS-Poll, criam uma matriz de terminais escondidos e posiciona os nós com problemas de terminais escondidos em diferentes grupos.

Neste capítulo, foram apresentados os principais conceitos necessários para o entendimento desta dissertação. Na Seção 2.2, foram apresentadas as características das camadas física e de enlace da norma IEEE 802.11ah e suas principais modificações em comparação às normas IEEE 802.11 anteriores. Nas Seções 2.3 e 2.4, foram explicadas as características do IEEE 802.11ah mais relevantes para a modelagem analítica desenvolvida neste trabalho. Na Seção 2.3 foi detalhado o funcionamento do mecanismo de janela de acesso restrito e na Seção 2.4 foi explicado o EDCA, protocolo de acesso aleatório ao meio utilizado no IEEE 802.11ah. A Seção 2.5 traz as características do IEEE 802.11ah implementadas no NS3 por Le Tian et al. [1] e, por fim, a Seção 2.5.3 são apresentados os principais trabalhos realizados na literatura sobre o IEEE 802.11ah.

3 MODELAGEM ANALÍTICA DO IEEE 802.11AH

3.1 INTRODUÇÃO

Ao observar semelhanças entre o IEEE 802.11 no modo de economia de energia e o IEEE 802.11ah, propõe-se o uso de uma cadeia de Markov de tempo discreto em duas dimensões baseada em nosso trabalho prévio desenvolvido para modelagem da operação do IEEE 802.11 no modo de economia de energia [10], e nos trabalhos de P. Swain [11] e [12]. No modo de economia de energia do IEEE 802.11, o tempo é dividido em intervalos de *beacon*, que é uma mensagem de sincronismo e que contém informações da rede enviado no início do intervalo de *beacon*, e um intervalo de *beacon* é divido em janela de anúncio de tráfego (ATIM, do inglês *Announcement Traffic Indication Map*) e janela de dados. A cada intervalo de *beacon*, as estações acordam ao escutar o *beacon* e trocam mensagens ATIM durante a janela de anúncio de tráfego caso tenham algum pacote para transmitir. As estações que trocam a mensagem ATIM com sucesso permanecem ativas durante a janela de dados, e as estações que não têm nenhum pacote para transmitir voltam ao estado inativo para economizar energia. De forma semelhante, o IEEE 802.11ah define a divisão do tempo em compartimento denominados "intervalos de *beacon*" e as estações transitam do estado ativo para o estado inativo constantemente.

No IEEE 802.11ah, o *beacon* contém o elemento RPS (do inglês *RAW Parameter Set*) que traz as informações do RAW, como a duração do slot RAW, número de slots RAW, critério de agrupamento das estações, entre outras. As estações acordam ao escutar o *beacon* e ficam ativas somente em seus slots RAW atribuídos, conforme as informações contidas no RPS. Dentro do slot RAW, no qual a estação foi atribuída, ela disputa pelo canal via a função de coordenação distribuída (DCF, do inglês *distributed coordination function*), onde está definido o protocolo CSMA/CA para acesso via contenção ao canal. Assim como no padrão IEEE 802.11, se o canal estiver ocioso por por um intervalo de tempo maior ou igual a DIFS segundos, a estação transmite seu pacote. Caso o canal esteja ocupado, a estação continua monitorando o canal até que ele esteja ocioso por um intervalo de tempo maior ou igual a DIFS, deste modo, a estação gera um intervalo de recuo (*backoff*) antes de transmitir, sendo este o recurso de prevenção de colisão do protocolo. Além disso, uma estação deve aguardar um tempo de *backoff* aleatório entre duas transmissões consecutivas, mesmo que o canal esteja ocioso um período de tempo maior ou igual a DIFS. Nesta dissertação, assumimos o modo básico do DCF, sem envio de quadros de controle RTS/CTS. Assim, ao transmitir um pacote de dados e ocorrer uma colisão (indicada pelo não recebimento da confirmação via quadro ACK de controle), a estação pode tentar retransmitir o pacote no próximo estágio de retransmissão, enquanto houver tempo hábil no slot RAW ou até atingir o número máximo de retransmissões definido pelo padrão. Ao término do slot RAW, as estações que pertencem a este slot irão para o estado inativo e acordarão novamente no próximo intervalo de *beacon*.

O modelo matemático parte do comportamento de cada nó em seu processo de *backoff*, e baseia-se em um processo de Markov com espaço de estados discreto e finito. No caso, utilizamos uma cadeia bidimensional para representar o contador de *backoff*, em que as linhas da cadeia representam os diferentes estágios de retransmissão. Para cada estágio i há um número de estados determinado pelo tamanho da janela de contenção, conforme a Seção 2.4.2. O objetivo da modelagem com uma cadeia de Markov é, a partir da sua solução, encontrar as probabilidades estacionárias de transmissão e de colisão de um quadro de dados, que serão fundamentais para o cálculo da vazão da rede. Um aspecto importante deste trabalho é a tentativa de se modelar a interrupção da atividade da estação por ocasião do final do slot RAW reservado a ela. Em particular, tentamos introduzir na cadeia de Markov uma probabilidade referente ao término do slot RAW durante a atividade de *backoff* da estação. Deste modo, propomos duas expressões empíricas para a probabilidade de término do slot RAW a fim de modelar este comportamento da estação. Para simplificar o modelo, descrevemos o comportamento da estação somente quando está ativa dentro do slot RAW atribuído a ela, sendo que o período no qual a estação está fora do slot RAW, em seu estado inativo, tratado no cálculo da vazão posteriormente, como será detalhado na Seção 3.6.

Para as condições iniciais do modelo, assume-se que a rede opera com um número fixo n de estações que operam segundo o serviço básico de acesso ao meio, ou seja, sem uso dos pacotes de controle RTS/CTS ("*four-way handshake*") para evitar o problema de terminal escondido, e o CSB (do inglês *Cross Slot Boundary*) é desabilitado. Além disso, assume-se que todas as estações estão na condição de saturação, em que sempre possuem quadros de dados prontos para transmissão em suas fileiras de pacotes. O intervalo de *beacon* é de tamanho fixo e a duração do grupo RAW é igual a duração do intervalo de *beacon*, ou seja, cada intervalo de *beacon* tem somente um grupo RAW, e as estações são distribuídas uniformemente entre os slots dentro do RAW. O modelo também assume que, uma vez que a estação estiver dentro do slot atribuído, ela pode tentar o acesso ao canal e transmitir seus pacotes enquanto houver tempo hábil para transmissão no slot.

Nas Seções 3.2 e 3.3, apresentamos o modelo de cadeia de Markov, junto com as equações das probabilidades de transição e respectiva solução para as probabilidades estacionárias dos estados da cadeia. Nas Seções 3.4 e 3.5 são apresentadas as propostas para a probabilidade de término do slot e na Seção 3.6 é apresentado o cálculo da vazão. Após encontrar solução do modelo analítico, foram feitas algumas avaliações numéricas no software MATLAB para validar o modelo e compará-lo com as simulações do IEEE 802.11ah, desenvolvidas por Le Tian et al. [1], realizadas no simulador de redes NS3. A validação do modelo será apresentado no Capítulo 4.

3.2 MODELO ANALÍTICO DO SISTEMA

Ao iniciar o slot RAW, a estação que deseja transmitir um pacote de dados entra em processo de *backoff* segundo a Eq.(2.2). Se o canal estiver ocioso por um intervalo de tempo maior ou igual a DIFS segundos, a estação transmite seu pacote. Caso o canal esteja ocupado, quando o

canal voltar a ficar ocioso por um intervalo de tempo maior ou igual a DIFS após a transmissão, a estação gera um intervalo de recuo (*backoff*) e transmite seu pacote de dados quando o contador chega a zero. Se o canal for ocupado durante o decremento do contador de *backoff*, o contador congela e retorna a decrementar quando o canal estiver ocioso novamente. Em caso de colisão, a estação pode retransmitir seu pacote no próximo estágio de recuo, enquanto houver tempo suficiente no slot RAW para transmitir ou até atingir o estágio de retransmissão máximo, definido pela norma. Dado que o modelo analítico deseja descrever este comportamento de cada estação, considere que o estágio de *backoff* é representado pelo processo estocástico $s(t)$ no instante de tempo t, e o contador de *backoff*, dentro do slot RAW, é representado pelo processo estocástico $b(t)$ também no instante de tempo t. A seguinte notação é utilizada para representar as probabilidades de transição dos estados da cadeia de Markov,

$$P\{(i_1, j_1)|(i_0, j_0)\} = P\{s(t+1) = i_1, b(t+1) = j_1 | s(t) = i_0, b(t) = j_0\}. \tag{3.1}$$

Por se tratar de um processo sem memória, na cadeia de Markov a probabilidade de estar no estado atual $t+1$ depende somente do estado anterior t. Portanto, a Eq.(3.1) representa a probabilidade da estação estar no estado atual (i_1, j_1) dado que ela estava no estado (i_0, j_0).

A Figura 3.1 contém o diagrama da cadeia de Markov discreta bidimensional que representa o algoritmo de recuo exponencial binário de uma estação. O parâmetro p descreve a probabilidade de colisão no slot RAW. Neste modelo, assumimos que p é constante e independente do estágio de retransmissão, mas dependente do tamanho da rede. A probabilidade do slot terminar durante qualquer estado $b_{i,j}$ é representada pelo parâmetro q. Em nossas hipóteses empíricas, q pode variar somente com o estágio de *backoff* ou pode variar com o estágio de *backoff* juntamente com o número de estações contidas no slot RAW. O parâmetro g é a probabilidade de congelamento do contador de *backoff* quando a estação percebe que o canal foi ocupado e independe do estágio de recuo. Considerando W_0 a janela de contenção mínima, o tamanho da janela de contenção no estágio de retransmissão i é dado por $W_i = 2^i W_0$, sendo W_m o tamanho máximo da janela de contenção, que corresponde ao m-ésimo estágio. As probabilidades de transição da cadeia de Markov são dadas por

$$\begin{array}{lllll}
(1) & P\{(i,j)|(i,j+1)\} = (1-q)(1-g), & i \in [0,m], & j \in [0, W_i - 1] \\
(2) & P\{(i,j)|(i,j)\} = g(1-q), & i \in [0,m], & j \in [0, W_i - 1] \\
(3) & P\{(0,j)|(i,j)\} = \dfrac{q}{W_0}, & i \in [0,m], & j \in [0, W_i - 1] \\
(4) & P\{(0,j)|(i,0)\} = \dfrac{(1-p)(1-q)}{W_0}, & i \in [0,m], & j \in [0, W_0 - 1] \\
(5) & P\{(i+1,j)|(i,0)\} = \dfrac{p(1-q)}{W_i}, & i \in [0,m-1], & j \in [0, W_i - 1] \\
(6) & P\{(0,j)|(m,0)\} = \dfrac{p(1-q)}{W_0}, & j \in [0, W_0 - 1]
\end{array} \tag{3.2}$$

- A primeira equação indica que o contador do recuo decrementa com a probabilidade $(1-q)(1-g)$, ou seja, se o slot RAW não acabou e o canal está livre para decrementar o contador;

- A segunda equação descreve a probabilidade do contador regressivo do recuo permanecer no mesmo estado, ou seja, congelar o contador, porque o slot RAW não acabou ainda, mas o canal está ocupado com transmissão de pacote de outra estação;
- A terceira equação descreve a probabilidade do slot terminar durante qualquer estágio da cadeia de Markov. Assim, a estação retorna ao estado inicial com probabilidade q e tenta transmitir um novo pacote no próximo slot RAW atribuído, sendo o novo estágio do contador escolhido aleatoriamente entre W_0 estados;
- A quarta equação fornece a probabilidade de transição relativa ao caso em que o contador zera e um pacote de dados é transmitido com sucesso, permitindo então a transmissão de um novo pacote de dados porque o slot RAW ainda não acabou;
- A quinta equação indica que houve colisão e a estação vai para o próximo estágio de retransmissão;
- A sexta equação descreve a probabilidade de transição quando ocorre uma colisão no último estágio de retransmissão e a estação vai para o início com probabilidade $p(1-q)$, descarta o pacote de dados e inicia a transmissão de um novo pacote;

3.3 SOLUÇÃO DA CADEIA DE MARKOV

Nesta seção será apresentada a solução da cadeia de Markov que modela o contador de *backoff*, assim como seus estágios de retransmissão, para uma estação que deseja transmitir um pacote de dados. Sabendo que o modelo considera apenas o comportamento da estação dentro do slot RAW para o qual foi atribuído para competir pelo canal, considere $b_{i,j}$ como sendo a probabilidade de transição estacionária da cadeia de Markov, ou seja:

$$b_{i,j} = \lim_{t\to\infty} P\{s(t) = i, b(t) = j\},\ i \in [0, m],\ j \in [0, W_i - 1]. \tag{3.3}$$

Como a cadeia de Markov apresentada é regular, a solução é dada por

$$b_{i,j} = \begin{cases} \dfrac{M}{W_0(1-g(1-q))}, & i = 0, \quad j = W_0 - 1 \\ \dfrac{M}{W_0(1-g(1-q))} \times \sum\limits_{l=0}^{W_0-(j+1)} (1-q)^l, & i = 0, \quad j \in [0, W_0 - 2] \\ \dfrac{p(1-q)}{W_i(1-g(1-q))}, & i \in [1, m], \quad j = W_i - 1 \\ \dfrac{p(1-q)}{W_i(1-g(1-q))} \times \sum\limits_{l=0}^{W_i-(j+1)} (1-q)^l b_{i-1,0}, & i \in [1, m], \quad j \in [0, W_i - 1], \end{cases} \tag{3.4}$$

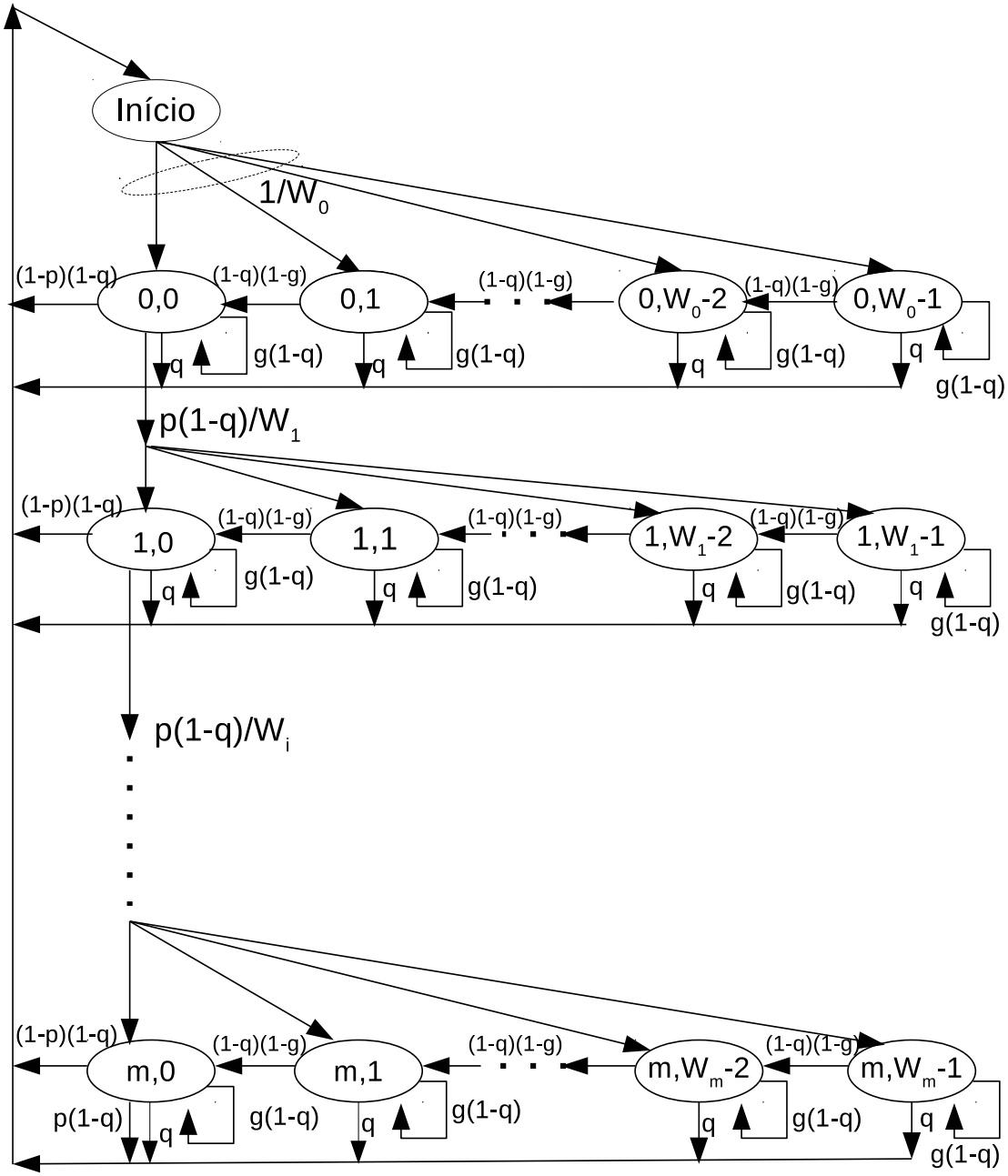

Figura 3.1: Cadeia de Markov para a modelagem da operação de recuo binário exponencial de uma estação segundo a norma IEEE 802.11ah.

em que M é dado por

$$M = q \sum_{i=0}^{m} \sum_{j=0}^{W_i-1} b_{i,j} + (1-p) \sum_{i=0}^{m} b_{i,0} + p(1-q) b_{m,0}.$$

Desejamos encontrar a probabilidade τ da estação transmitir um pacote de dados. Pela cadeia de Markov, a estação transmite um pacote quando atinge o estado $b_{i,0}$. Portanto, a probabilidade da estação transmitir um pacote de dados, em qualquer instante de tempo, é dada por

$$\tau = \sum_{i=0}^{m} b_{i,0} = \sum_{i=0}^{m} \left[\frac{p(1-q)}{W_i(1-g(1-q))} \sum_{l=0}^{W_i-1} (1-q)^l b_{i-1,0} \right]. \tag{3.5}$$

Como podemos observar, a probabilidade de transmissão τ depende da probabilidade de colisão p, da probabilidade do canal estar ocupado g, e da probabilidade q do slot RAW finalizar, as quais ainda são desconhecidas. Um pacote colide com outro se ao menos uma das estações $n-1$ restantes também transmitir um pacote de dados durante o slot. Assumindo que as transmissões de cada estação são independentes, a probabilidade de colisão no slot RAW é dada por

$$p = 1 - (1-\tau)^{(n-1)}, \tag{3.6}$$

em que n é o número de estações da rede.

A probabilidade g do canal estar ocupado, assim como a probabilidade de colisão, também é a probabilidade de haver alguém transmitindo no canal. Portanto a probabilidade g é dada por

$$g = p = 1 - (1-\tau)^{(n-1)}, \tag{3.7}$$

Dado que o valor de p depende de τ, assim como o valor de τ depende de p, as Eqs.(3.5) e (3.6) formam um sistema não linear com duas incógnitas p e τ, cuja solução é obtida por meio de cálculo numérico com um ponto fixo de iteração. Logo, probabilidade estacionária $b_{0,0}$, assim como as probabilidades estacionárias dos outros estados, são encontradas a partir da solução do sistema juntamente com a condição de normalização da cadeia de Markov, dada por

$$1 = \sum_{i=0}^{m} \sum_{j=0}^{W_i-1} b_{i,j}. \tag{3.8}$$

Para finalizar a solução da cadeia de Markov, ainda falta calcular a probabilidade q de término do slot RAW. Possivelmente, a probabilidade q depende do número de estações contidas no slot, pois, quanto maior o número de estações competindo pelo canal simultaneamente, maior o tempo gasto no processo de *backoff*, portanto maior será a probabilidade de término do slot q. Também, possivelmente, espera-se que a probabilidade q dependa do estágio de *backoff* o qual a estação está, pois à medida que o estágio de *backoff* avança, maior a probabilidade do slot RAW

terminar. Com a possível dependência de q pelo número de estações no slot, observamos que esta probabilidade também pode depender da probabilidade de transmissão τ, uma vez que τ representa a probabilidade de uma estação transmitir. Entretanto, dada a complexidade de modelar q incluindo esta dependência da probabilidade de transmissão, faremos aproximações empíricas sobre seu cálculo nas Seções 3.4 e 3.5.

3.4 PROBABILIDADE DE TÉRMINO DO SLOT COMO FUNÇÃO DO ESTÁGIO DE RECUO

No IEEE 802.11ah, a estação compete pelo canal no slot atribuído a ela, e vai para o estado inativo ao término do seu slot RAW. Entretanto, o modelo apresentado para a cadeia de Markov descreve um comportamento dinâmico da estação em estado estacionário, em que a estação sempre está no estado ativo tentando o acesso ao canal. Na tentativa de representar esse momento de término do slot, introduzimos ao modelo a probabilidade q, que descreve a probabilidade de término do slot RAW.

Como uma primeira proposta para a expressão desta probabilidade, observamos que, em princípio, o tempo que a estação gasta em cada estágio de *backoff* deve crescer, em média, com o tamanho da janela de contenção (afirmamos ser em média porque o valor sorteado para o contador é sempre escolhido aleatoriamente entre $[0, W_i - 1]$). Logo, à medida que o número de tentativas de retransmissão aumenta, mais tempo é consumido pela estação em um slot RAW específico. Consequentemente, à medida que o número de estágios avança, a chance do slot RAW acabar antes da estação finalizar suas tentativas de retransmissão aumenta. Portanto, para representar o aumento desta probabilidade de término de slot com o número de estágios percorridos, propomos uma probabilidade de término de slot RAW que cresce proporcionalmente com o número k de estágios percorridos, ou seja,

$$q_k = \frac{k}{m+1},\ k \in [0, m]\ , \tag{3.9}$$

em que m indica o número máximo de estágios e assumimos que no primeiro estágio a chance de término do slot RAW é $q_0 = 0$.

3.5 PROBABILIDADE DE TÉRMINO DO SLOT COMO FUNÇÃO DO ESTÁGIO DE RECUO E NÚMERO DE ESTAÇÕES

Após realizar os resultados do modelo utilizando a probabilidade q_k descrita na Seção 3.4, e compará-los com simulações no NS3, percebemos que a probabilidade de término da slot RAW deve ser menor à medida que o número de slots dentro do RAW aumenta e o número de estações a ser distribuído entre os slots permanece fixo. Isto acontece porque o número de estações contidas

no slot RAW diminui quando o RAW é dividido em mais slots. Assim a estação gasta menos tempo no processo de *backoff* quando tem menos competição pelo canal, e a probabilidade do slot RAW acabar durante uma tentativa de transmissão de pacote de dados tende a diminuir. Portanto, propomos uma nova expressão empírica para a probabilidade q em que ela depende não apenas do estágio retransmissão k, mas também do número de estações contidas no slot RAW.

Vamos assumir que, se fosse justo, o IEEE 802.11 alocaria uma fração de tempo igualitária para todas as n estações contidas em um slot RAW. Assim, neste caso, cada estação receberia, em princípio, $1/n$ do tempo total do slot RAW para o acesso ao canal. Portanto, ao final do uso desta sua "fração de tempo", o slot RAW estaria finalizado para esta estação. Neste caso, com probabilidade $1 - 1/n$ a estação terá terminada a sua participação no slot RAW. Deste modo, a probabilidade q_k^n pode ser expressada por

$$q_k^n = P[\text{estação está no estágio } k, \text{ fração do slot dividida entre } n \text{ nós finaliza}], \tag{3.10}$$

que pode ser reescrito como

$$q_k^n = P[\text{fração do slot dividida entre } n \text{ nós finaliza} \mid \text{estação está no estágio } k] \times P[\text{estação está no estágio } k], \tag{3.11}$$

Como a probabilidade da estação estar no estágio k é dada pela Eq.(3.9), podemos então rescrever como

$$q_k^n = \left(1 - \frac{1}{n}\right)\left(\frac{k}{m+1}\right), \tag{3.12}$$

ou seja,

$$q_k^n = \left(\frac{n-1}{n}\right)\left(\frac{k}{m+1}\right). \tag{3.13}$$

Na Seções 4.3, 4.4, 4.5 e 4.6, avaliamos os resultados do modelo matemático com o uso das Eq.(3.9) e Eq.(3.13), e comparamos com os resultados via simulações computacionais a fim de verificar qual expressão de q melhor representa o término do slot no IEEE 802.11ah. Finalmente, vale salientar que a proposição de uma expressão empírica para a probabilidade de término de slot RAW q não é uma tarefa fácil. Por exemplo, para entender esta dificuldade, P. Swain et al. [11] propuseram o uso de uma probabilidade de término de slot para representar o término de slot ATIM no modo de economia de energia do IEEE 802.11. No entanto, dada a dificuldade em obter uma expressão, os autores propuseram uma probabilidade baseada em resultados de simulação computacional. Ou seja, em vez de tentar uma expressão analítica, os autores realizaram simulações para tentar encontrar valores numéricos específicos para determinados cenários. Abordagem parecida vai ser também apresentada na Seção 4.7 para entendermos o impacto do parâmetro q e possível fator de escala necessário para uma melhor aproximação do modelo com as simulações.

3.6 CÁLCULO DA VAZÃO MÉDIA POR ESTAÇÃO E VAZÃO AGREGADA DA REDE

Nesta seção, apresentamos o cálculo da vazão média por slot RAW, assim como a vazão média por intervalo de beacon para um dado slot RAW e, a partir deste cálculo, a vazão agregada da rede considerando todos os slots RAW dentro de um intervalo de beacon. Seja S_{DADOS} a vazão média observada dentro de um slot RAW, definida como a razão entre o tamanho médio de um pacote de dados (em bits) transmitido com sucesso, pelo tempo médio de um "slot" no contexto da operação em tempo discreto da função de coordenação distribuída (DCF) [25]. No caso, para o DCF, o tempo é dividido em mini-slots, e em cada mini-slot, podemos observar apenas três estados para o canal: ocioso, com transmissão bem-sucedida de pacote de dados, ou com colisão de pacotes de dados. O comprimento deste slot de tempo é definido pela norma e depende da camada física. Assim, a vazão por slot RAW será calculada por

$$S_{DADOS} = \frac{\text{E[tamanho do pacote de dados em bits transmitido em um slot DCF]}}{\text{E[duração de um slot DCF]}}. \tag{3.14}$$

Para encontrarmos o comprimento médio de um slot (no contexto da operação do DCF), precisamos considerar cada um dos estados possíveis do canal. Seja P_{tr} a probabilidade de existir pelo menos uma transmissão de pacotes no slot considerado, dado que n estações disputam pelo canal com probabilidade τ. Como cada estação transmite independente da outra, temos que

$$P_{tr} = 1 - (1 - \tau)^n. \tag{3.15}$$

A probabilidade de sucesso P_s é a probabilidade de exatamente uma estação transmitir no canal, condicionada ao fato que pelo menos uma transmissão aconteceu no slot, portanto,

$$P_s = \frac{n\tau(1-\tau)^{(n-1)}}{P_{tr}}. \tag{3.16}$$

Deste modo, vazão por slot RAW é dada por

$$S_{DADOS} = \frac{P_s P_{tr} E[P]}{(1 - P_{tr})\sigma + P_s P_{tr} T_s + (1 - P_s) P_{tr} T_c} \ , \tag{3.17}$$

em que $E[P]$ é o tamanho médio de um pacote de dados, $P_s P_{tr}$ é a probabilidade de um pacote ser transmitido com sucesso em um slot DCF, $(1 - P_{tr})$ é a probabilidade do slot DCF estar desocupado e $(1 - P_s)P_{tr}$ é a probabilidade de haver colisão entre pacotes de dados no slot DCF. Também seja T_s o tempo em que o canal está ocupado com uma transmissão com sucesso, T_c o tempo em que o canal está ocupado com uma transmissão com colisão e σ o intervalo de tempo

em que o slot DCF está vazio. Os valores de T_s e T_c são dados por

$$T_s = DIFS + \frac{H}{\text{Taxa básica}} + \frac{E[P]}{\text{Taxa de dados}} + 2\delta + SIFS + ACK \tag{3.18}$$

e

$$T_c = DIFS + \frac{H}{\text{Taxa básica}} + \frac{E[P]}{\text{Taxa de dados}} + SIFS + ACK_{TO} \ , \tag{3.19}$$

em que taxa básica é a taxa de transmissão utilizada para transmissão do cabeçalho do pacote de dados representado por H, incluindo tanto o cabeçalho da camada física como o cabeçalho da camada de enlace. Os valores dos intervalos de tempo DIFS e SIFS, assim como o tempo para transmissão de um quadro de controle ACK são os valores especificados na norma IEEE 802.11ah e δ é o atraso de propagação do canal, também especificado na norma. O valor de ACK_{TO} é o período tempo que a estação deve aguardar após uma colisão e é dado por

$$ACK_{TO} = 2\delta + SIFS + ACK. \tag{3.20}$$

A cadeia de Markov proposta representa somente o comportamento da estação dentro do slot RAW. Portanto, a Eq.(3.17) é a vazão apenas dentro do slot RAW. A fim de incluir no modelo o impacto do tempo no qual a estação está no período inativo, a vazão média total em um intervalo de *beacon*, considerando somente um slot RAW, é dada por

$$S_{BEACON} = S_{DADOS} \times \frac{\text{Tamanho do slot RAW}}{\text{Duração do Intervalo de Beacon}}. \tag{3.21}$$

Considerando que no modelo proposto e nas simulações computacionais estamos considerando que o *Cross Slot Boundary* (CSB) está desabilitado, o tamanho real do slot RAW é igual à duração do slot RAW menos o tempo de *holding* e o tempo de guarda. Para tempo de *holding*, a estação deve verificar se há tempo suficiente no slot RAW para realizar uma transmissão com sucesso. Caso contrário, a estação deve abster-se de transmitir. Portanto, consideramos o tempo de *holding* igual ao tempo para uma transmissão com sucesso, ou seja, $T_h = T_s$. O tempo de guarda T_g é o intervalo de tempo entre os slots RAW para que, caso o CSB esteja habilitado, a transmissão de um slot RAW não sobreponha outro slot RAW. Sendo assim, a vazão por intervalo de *beacon* ajustada é dada por

$$S_{BEACON} = S_{DADOS} \times \frac{\text{Tamanho do slot RAW} - T_h - T_g}{\text{Duração do Intervalo de Beacon}}. \tag{3.22}$$

Finalmente, para obtermos a vazão agregada da rede, ou seja, a vazão resultante da transmissão de pacotes de dados em todos os slots RAW definidos dentro do intervalo de beacon, devemos somar as vazões obtidas em todos os slots RAW. Observe que, dependendo da alocação do número de estações dentro de cada slot RAW, alguns slots RAW podem ter mais ou menos estações que outros slots RAW. Logo, dada uma distribuição qualquer de estações dentro de cada slot RAW, a

vazão média por intervalo de *beacon* de toda a rede será a soma das vazões calculadas segundo a Eq.(3.22) para cada número de estações em cada slot RAW. É importante enfatizar que, em simulações computacionais, a vazão da rede é calculada considerando o número total de pacotes recebidos (e portanto, de bits) dentro de um intervalo de tempo de operação da rede. Logo, caso estivéssemos interessados na vazão da rede do ponto de vista de um único slot RAW (ex: todas as estações estão alocadas em um único slot RAW), os pacotes da redes seriam todos transmitidos apenas quando ocorresse o slot RAW específico em cada intervalo de beacon. Portanto, considerando o tempo total de operação da rede, a vazão corresponderia ao número total de pacotes recebidos apenas no slot RAW específico, dividido pelo tempo total de operação da rede (ou seja, a soma dos intervalos de beacon).

Neste capítulo, apresentamos o modelo analítico do IEEE 802.11ah desenvolvido nesta dissertação. Foi proposto o uso de uma cadeia de Markov de tempo discreto bidimensional baseada em nosso trabalho prévio, em que foi proposta uma cadeia de Markov para o IEEE 802.11 no modo de economia de energia. Esta cadeia de Markov modelou o comportamento de uma estação em processo de *backoff* dentro do seu slot de atividade. Na Seção 3.2 foi apresentado o modelo analítico do sistema e na Seção 3.3 foi apresentada a solução deste modelo teórico. A fim de introduzir ao modelo analítico o momento em que a estação vai do estado ativo para o estado inativo ao término de seu slot de atividades, propomos duas expressões empíricas para a probabilidade de término do slot nas Seções 3.4 e 3.5. Por fim, apresentamos o cálculo da vazão média por slot RAW e a vazão agregada da rede na Seção 3.6.

4 AVALIAÇÃO DE DESEMPENHO

4.1 INTRODUÇÃO

Neste capítulo, apresentamos resultados da vazão média agregada da rede para diferentes números de estações e para diferentes números de slots RAW por intervalo de beacon.

Nas avaliações, apresentamos os resultados para as duas abordagens assumidas para o cálculo empírico da probabilidade de término de slot q, e avaliamos o desvio percentual de predição do modelo teórico frente aos resultados de simulação.

Adicionalmente, no final do capítulo, apresentamos os resultados para o modelo teórico quando a probabilidade de término de slot é ajustada segundo uma constante multiplicadora obtida diretamente a partir dos resultados de simulação. Tal procedimento segue a abordagem adotada por Swain et al. [11] que propuseram tal aproximação para o cálculo de término de slot ATIM no contexto do modo de economia de energia do IEEE 802.11. Apresentamos estes resultados para entendermos o efeito da probabilidade q e para prover subsídios para propostas futuras de cálculo desta probabilidade fundamental.

O restante deste capítulo está estruturado da seguinte forma. A Seção 4.2 apresenta o cenário considerado no modelo e nas simulações, assim como os parâmetros utilizados. As Seções 4.3, 4.4, 4.5 e 4.6 apresentam a análise do modelo utilizando as propostas de q. Por fim, a Seção 4.7 apresenta valores de constantes os quais, multiplicando pelo q que depende somente do estágio de recuo, é possível tornar o modelo mais próximo da simulação.

4.2 CENÁRIO CONSIDERADO

Para a realização das simulações computacionais, tanto do modelo no MATLAB, quanto na implementação do IEEE 802.11ah no NS3, assumimos que a rede opera com um número fixo n de estações, com tráfego saturado, isto é, que cada estação sempre tem um pacote de dados pronto para transmissão em sua fila, e sem utilizar os pacotes de controle RTS/CTS. Cada intervalo de *beacon* possui um grupo RAW com a mesma duração do intervalo de beacon, ou seja, apenas um grupo RAW é simulado, mas com diferentes números de slots RAW internos. As estações são distribuídas uniformemente entre os slots. No entanto, vale ressaltar que nem sempre todos os slots RAW têm o mesmo número de estações alocadas. Por exemplo, se n = 80 e existem 15 slots RAW. Neste caso, 10 slots RAW terão 5 estações e 5 slots RAW terão 6 estações. O *Cross Slot Boundary* foi desabilitado, o que significa que nenhuma transmissão de pacote de dados com sucesso pode ultrapassar o fim de um slot RAW. As estações estão posicionadas aleatoriamente em volta de um ponto de acesso AP (do inglês *Access Point*), cuja área é um círculo de raio

Tabela 4.1: Tabela de parâmetros

Parâmetro	**Valor**
Tamanho do pacote de dados	256 bytes
ACK	14 bytes + Cabeçalho da Camada Física
Cabeçalho da Camada Física	192μs
Cabeçalho da Camada de Enlace	34 bytes
Taxa de Transmissão Básica	1 Mbps
Taxa de Transmissão de Dados	7,8 Mbps
Duração do slot	52μs
δ	$3{,}3\mu$s
SIFS	160μs
DIFS	1120μs
ACK_{TO}	$2 \times \delta + SIFS + ACK$
CW_{min}	16
CW^{d}_{max}	1024
Esquema de modulação e codificação	MCS8
Largura de banda no canal	2MHz
Intervalo de Beacon	0,1s

igual a 50 metros. Além disto, considerando um cenário IoT em que os dispositivos são de baixa potência e utilizam baterias como forma de energia, a potência de transmissão foi limitada a 0 dBm. Os parâmetros das camadas Física e MAC utilizados nas análises foram o mesmos parâmetros utilizados por Le Tian et al. [1] e são apresentados na Tabela 4.1.

Para a geração de tráfego no NS3, colocamos o intervalo entre geração de pacotes de dados via UDP igual a $0{,}002$ segundos, pois com este intervalo conseguimos o tráfego saturado na rede considerado. Para o modelo de perda de propagação do canal, modificamos o modelo que estava no módulo para um modelo de canal ideal sem perdas de propagação no espaço livre, uma vez que consideramos uma camada física ideal no modelo analítico e os estudos desta dissertação envolvem apenas a camada de enlace. Os outros parâmetros são alterados por meio valores passados na de linhas de comando via terminal, como o número de estações na rede, o número de grupos RAW e o número de slots RAW, conforme cada caso considerado. Foram realizadas 10 simulações diferentes com valores de sementes gerados aleatoriamente para cada ponto dos gráficos. Os gráficos apresentados correspondem aos valores médios da vazão agregada da rede para cada caso estudado. Em seguida, utilizamos esses mesmos parâmetros para obter os resultados numéricos do modelo teórico no MATLAB, para os dois casos da probabilidade q e comparamos com as simulações no NS3. Lembrando que a vazão agregada do modelo foi obtida multiplicando os resultados da Eq. (3.22) pelo número de slots em cada caso. Nas seções a seguir, apresentamos as análises para os casos de 2, 5, 10 e 15 slots RAW.

4.3 ANÁLISE DA VAZÃO COM 2 SLOTS RAW

Nesta seção, vamos analisar o caso em que o RAW foi dividido em 2 slots. A Figura 4.1 contém os resultados da vazão média agregada (sobre todos os slots RAW) para diferentes valores no número total de estações para o caso em que são utilizados 2 slots RAW. As estações foram divididas uniformemente entre os slots. Entretanto, para o caso com 5 estações, consideramos no modelo teórico que um slot possui 2 estações e o outro slot possui 3 estações. Calculamos as vazões médias por intervalo de *beacon* para cada slot RAW e depois somamos os resultados no cálculo da vazão agregada. A vazão agregada nos outros valores de número de estações foram obtidas multiplicando a vazão da Eq. (3.22) por 2 slots.

Na Figura 4.1, a curva em preto é a vazão do modelo com a probabilidade de término do slot q dependendo apenas do estágio de recuo o qual a estação se encontra. Na legenda, este caso está identificado como q_k. A curva em azul é a vazão do modelo com a probabilidade q_k^n que depende do estágio de recuo e do número de estações contidas no slot RAW, indicada na legenda como q_k^n. Por fim, a curva em vermelho apresenta a vazão média obtida das simulações no NS3. Podemos observar um comportamento decrescente da vazão da rede saturada com o RAW dividido em 2 slots à medida que o numero total de estações na rede aumenta. Este comportamento é devido ao aumento da probabilidade de colisão entre os pacotes dados quando o número de estações dentro do slot RAW aumenta. As curvas do modelo analítico apresentam o comportamento semelhante à curva da simulação. Entretanto, apresentam discrepâncias quanto aos valores da vazão média, sendo que não houve quase nenhuma diferença entre os casos q_k e q_k^n. Observa-se também que, apesar da taxa de transmissão de dados estar configurada em $7,8$ Mb/s, a vazão máxima do sistema é aproximadamente 1 Mb/s. Isto acontece porque o pacote de dados é relativamente pequeno, e o canal também é ocupado pelas transmissões de cabeçalhos e pacotes de controle, as quais ocorrem a 1 Mb/s. Portanto, a transmissão de cabeçalho e pacotes de controle influenciam de forma significativa a vazão total do sistema, e isso ocorre em todos os casos de número de slots RAW analisados neste trabalho.

Na Figura 4.2 estão os desvios percentuais que correspondem às discrepâncias entre o modelo analítico e a simulação NS3. O modelo considerando a probabilidade q_k e o modelo considerando a probabilidade q_k^n apresentam erros iguais para quase todos os valores de n (número de estações na rede), exceto para $n = 5$, $n = 20$, $n = 30$ e $n = 40$ estações, em que o modelo com q_k^n apresenta resultados um pouco melhores em relação ao modelo com q_k. Sendo assim, não houve diferença significativa entre as propostas de probabilidades de término de slot q_k e q_k^n quando o RAW é dividido em 2 slots.

4.4 ANÁLISE DA VAZÃO COM 5 SLOTS RAW

Nesta seção, consideramos o caso de 5 slots dentro do RAW. A Figura 4.3 contém os resultados da vazão média agregada (sobre todos os slots RAW) para diferentes valores no número total de

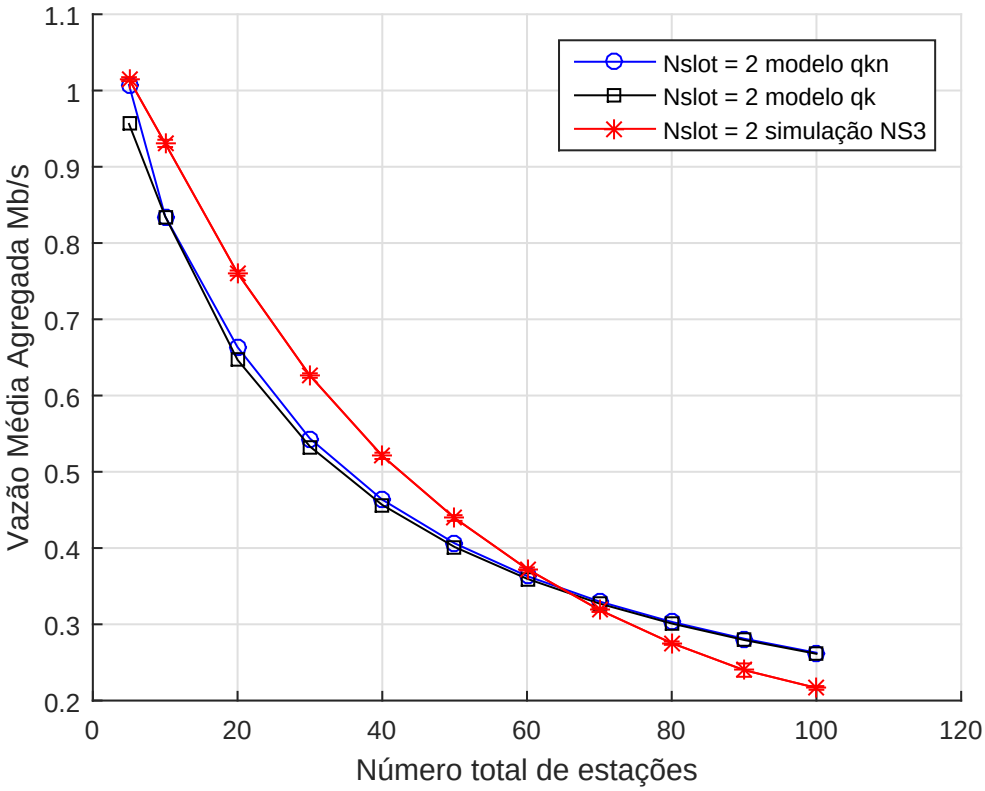

Figura 4.1: Comparação modelo analítico com simulação no NS3 com 2 slots RAW.

estações para o caso em que são utilizados 5 slots RAW. A figura mostra os resultados numéricos com as probabilidades q_k e q_k^n e de simulação. As estações foram divididas igualmente entre os slots e, para obter a vazão agregada do sistema, bastou multiplicar a vazão da Eq. (3.22) por 5, que são os 5 slots.

Na Figura 4.3, cujos gráficos são com 5 slots RAW, observa-se ainda uma curva decrescente, porém houve um aumento na vazão da rede com relação aos gráficos com 2 slots RAW, principalmente quando o número de estações na rede aumenta. Ao dividir as estações em mais grupos, haverá menos competição pelo canal no intervalo de tempo determinado ao slot. Portanto, a probabilidade de colisão diminui, levando a um aumento da vazão. A vazão obtida com modelo analítico apresenta um comportamento semelhante ao obtido por meio das simulações no NS3, sendo que, para o caso de 5 slots RAW, o modelo com a probabilidade q_k^n aproximou-se um pouco mais da simulação.

A Figura 4.4 apresenta um gráfico de desvio percentual para o caso de 5 slots e mostra que o modelo com a probabilidade q_k^n realmente apresenta resultados melhores em comparação ao modelo com a probabilidade q_k. O erro percentual do modelo com q_k^n em relação à simulação no NS3 foi menor que o erro do modelo com q_k em todos os valores de n considerados.

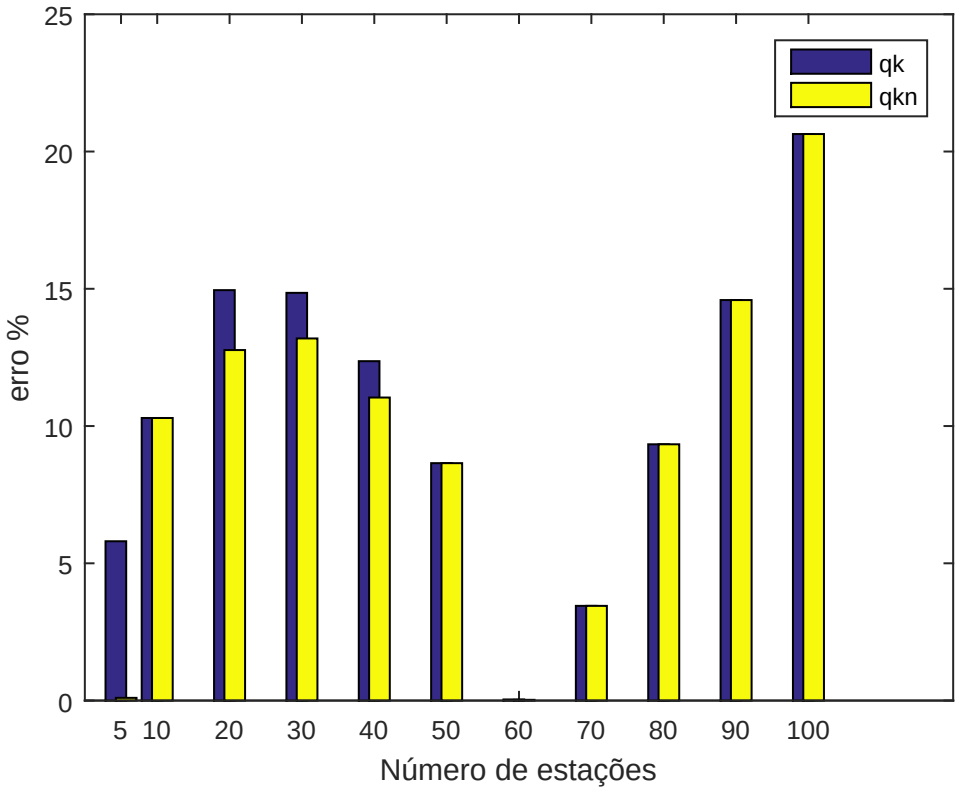

Figura 4.2: Gráfico de barras com erro percentual entre modelo analítico e simulação no NS3 para 2 slots.

4.5 ANÁLISE DA VAZÃO COM 10 SLOTS RAW

Vamos considerar agora o caso de 10 slots RAW. As estações foram divididas uniformemente entre os slots. Para o caso com 5 estações na rede, apenas 5 slots RAW serão ocupados por uma estação e os outros 5 slots RAW ficarão vazios. Assim, a vazão agregada para este caso, é a soma das vazões médias dos 5 slots ocupados. A Figura 4.5 contém os resultados da vazão média agregada para o caso de 10 slots RAW.

O caso com 10 slots (Figura 4.5), apresenta um comportamento diferente dos anteriores. Com apenas $n = 5$ estações na rede, há slots vazios. Sendo assim, o canal não será totalmente aproveitado durante o RAW. Por isso, a vazão inicia com um valor mais baixo, aumenta com $n = 10$ estações e começa a decrescer com o aumento de estações na rede, pois a probabilidade de colisão também aumenta. O modelo teórico apresenta um comportamento semelhante à simulação, sendo bem preciso nos pontos em $n = 5$ e $n = 10$, poderia-se dizer que $n = 20$ e $n = 30$ também, com a probabilidade q_k^n. Porém, a vazão do modelo decresce mais rápido em relação à vazão da simulação, indicando que o modelo teórico está mais conservador com relação às perdas na vazão.

Na Figura 4.6, o modelo com a probabilidade q_k^n possui o desvio percentual menor em comparação ao modelo com a probabilidade q_k em todos os valores de n. Percebe-se que o modelo com a probabilidade q_k^n vem apresentando mais proximidade ao gráfico da simulação no NS3 à

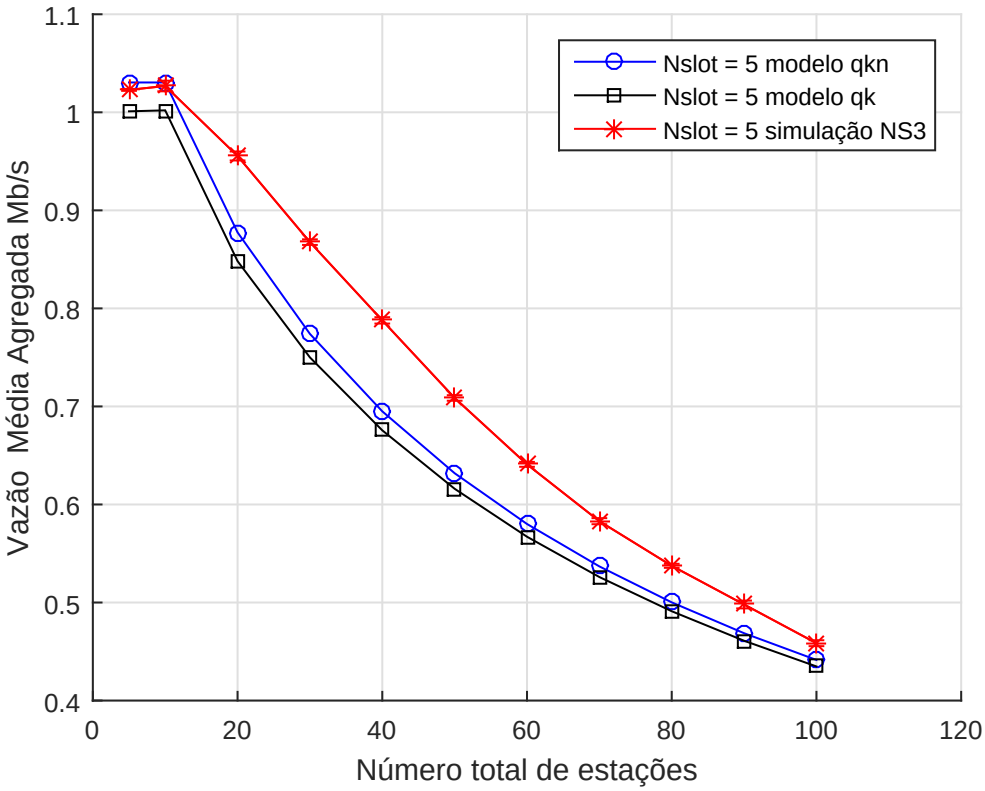

Figura 4.3: Comparação modelo analítico com simulação no NS3 com 5 slots RAW.

medida que o número de slots RAW aumenta. Ao incorporar o número de estações por slot à expressão de q, estamos considerando que o número de estações por slot diminui ao aumentar a quantidade de slots dentro do RAW, reduzindo a competição pelo canal. Assim, a probabilidade que o slot termine durante uma transmissão também diminui, pois a estação ficará menos tempo no processo de *backoff*.

4.6 ANÁLISE DA VAZÃO COM 15 SLOTS RAW

Finalmente, consideramos o caso de 15 slots RAW. As estações foram divididas uniformemente entre os slots, mas, como há um número de estações n que não são divisíveis por 15, distribuímos uniformemente entre os slots RAW até o menor número divisível mais próximo de n e depois distribuímos as estações restantes entre os slots RAW até que todas as estações sejam atribuídas a um slot RAW. Desta forma, os slots RAW terão números de estações diferentes. Por exemplo, para $n = 50$, 5 slots terão 4 estações e 10 slots terão 3 estações. A Figura 4.7 contém os resultados da vazão média agregada para o caso de 15 slots RAW.

Na Figura 4.7, observa-se que o modelo analítico foi bastante discrepante em comparação a simulação no NS3 para o caso de 15 slots RAW. Observamos por meio das simulações no NS3 que, ao dividir o RAW em mais slots, haverá menos dispositivos competindo pelo canal dentro

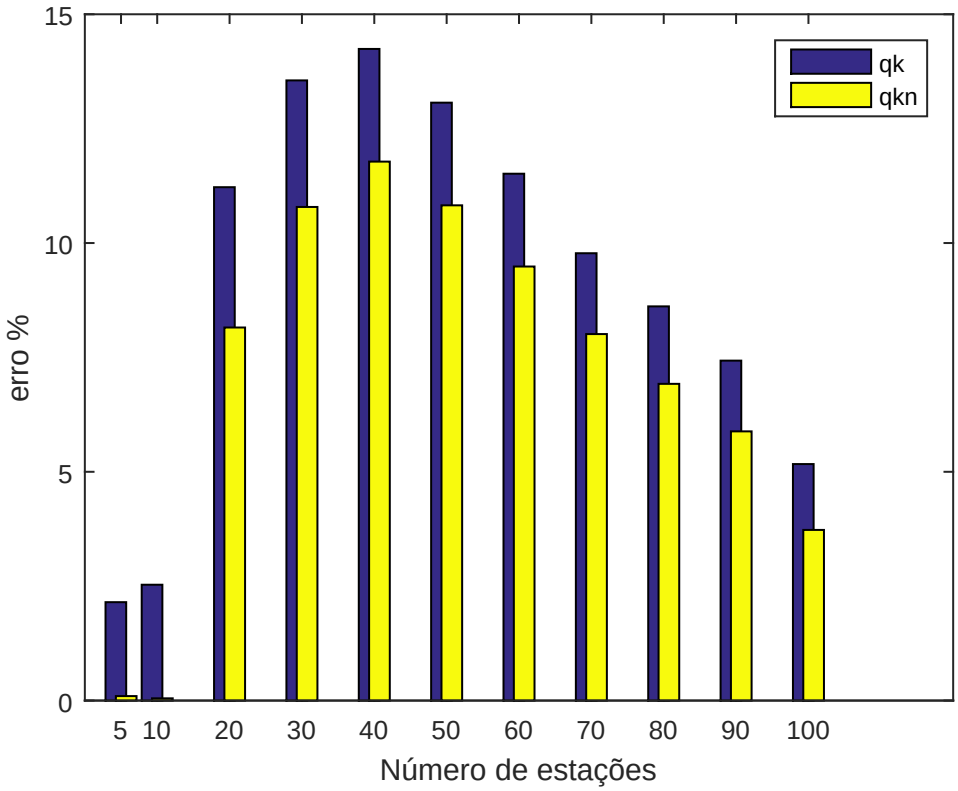

Figura 4.4: Gráfico de barras com erro percentual entre modelo analítico e simulação no NS3 para 5 slots.

de cada slot, o que faz com que a vazão total da rede aumente (observando também que o slot deve ter tempo suficiente para a transmissão de pelo menos um pacote, caso contrário, a estação não terá tempo hábil para transmitir seu pacote e a vazão diminuirá). Entretanto, o modelo está mais pessimista e conservador quanto ao que de fato acontece. Dado que a duração do RAW é fixa, a janela de atividades diminui ao dividir o RAW em mais slots e, como o modelo analítico representa o comportamento de uma rede em regime estacionário, a dinâmica das estações dentro do slot pode não estar sendo representada corretamente pelo modelo teórico devido à esta janela curta de atividades. Do mesmo modo que o caso com 10, a vazão inicia com o valor mais baixo pelo fato de haver slots vazios, cresce até $n = 20$ e decresce com o aumento do número de estações devido à maior competição pelo canal.

Como podemos ver na Figura 4.8, o modelo analítico realmente apresentou um grande desvio percentual para caso com 15 slots, sendo que o modelo utilizando a probabilidade q_k^n obteve um erro menor. No entanto, este cenário no qual o modelo falha não é um cenário esperado para Internet das Coisas. Em um cenário de Internet das Coisas, muitos dispositivos estão conectados a um único ponto de acesso, portanto, certamente haverá mais dispositivos competindo pelo canal em cada slot. Este cenário para 15 slots que estamos considerando está caminhando para um cenário TDMA, em que a estação possui o seu slot para transmitir sem disputar o acesso ao canal. Em todo caso, a natureza da discrepância do modelo para este caso precisa ser melhor estudada e entendida.

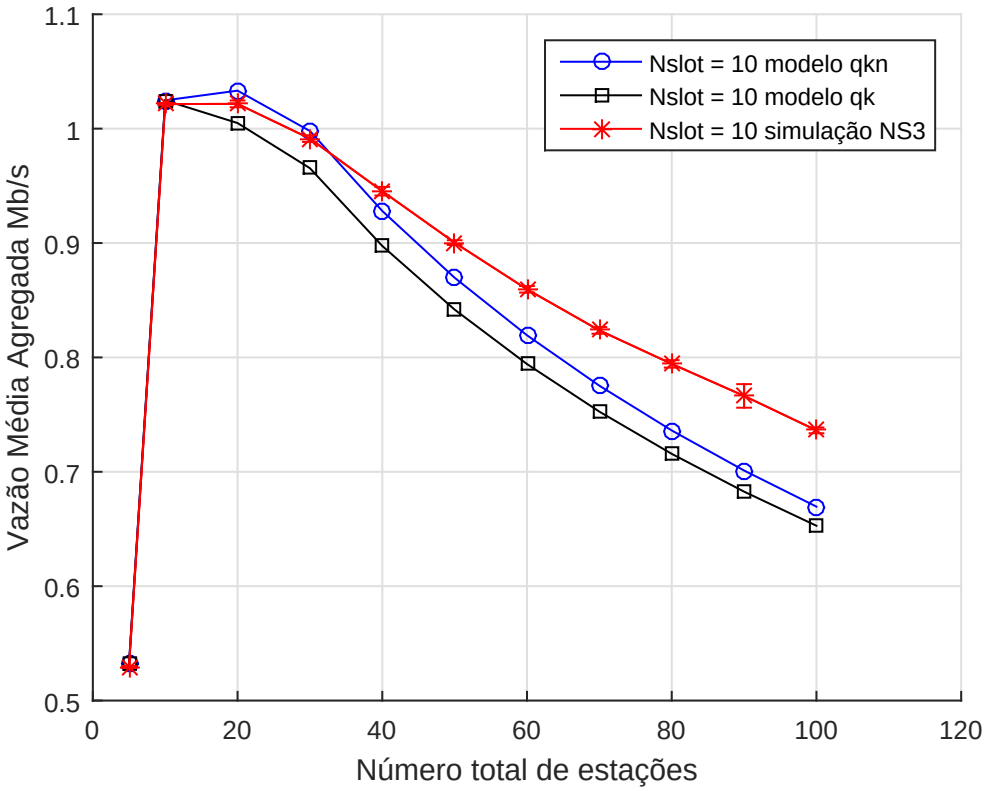

Figura 4.5: Comparação modelo analítico com simulação no NS3 com 10 slots RAW.

4.7 ANÁLISE DA VAZÃO COM PROBABILIDADE Q AJUSTADA POR UMA CONSTANTE C

P. Swain [11] e [12], modelou o processo de *backoff* de uma rede saturada baseada no modo de economia de energia do IEEE 802.11 com uma cadeia de Markov, e introduziu ao seu modelo a probabilidade de término do slot q, tanto para a janela de anúncio de tráfego, quanto para a janela de dados. Em seu modelo, ela considerou que q depende do número de estações na rede e do tamanho da janela. Contudo, não apresentou uma expressão analítica para esta probabilidade, mas obteve um valor a partir dos resultados das simulações. Para tal, ela observou quais valores de probabilidade de sucesso aproximavam o modelo teórico das simulações. Portanto, tomando como base os trabalhos de P. Swain [11] e [12], nesta seção, escolhemos valores da probabilidade de término do slot q que aproxime o nosso modelo analítico das simulações computacionais por meio dos resultados obtidos nas simulações.

Obter o valor numérico a partir de simulações não é o esperado de um modelo teórico. Entretanto, este estudo tem como objetivo entender o impacto da probabilidade de término do slot q no modelo e entender o comportamento de seu valor à medida que o número de slots RAW aumenta. Este estudo servirá para estudos futuros para a busca de expressões empíricas melhores para a modelagem da probabilidade de término so slot q. Deste modo, encontraremos o valor de uma constante c a partir dos valores das vazões obtidos nas simulações que, ao multiplicar a probabi-

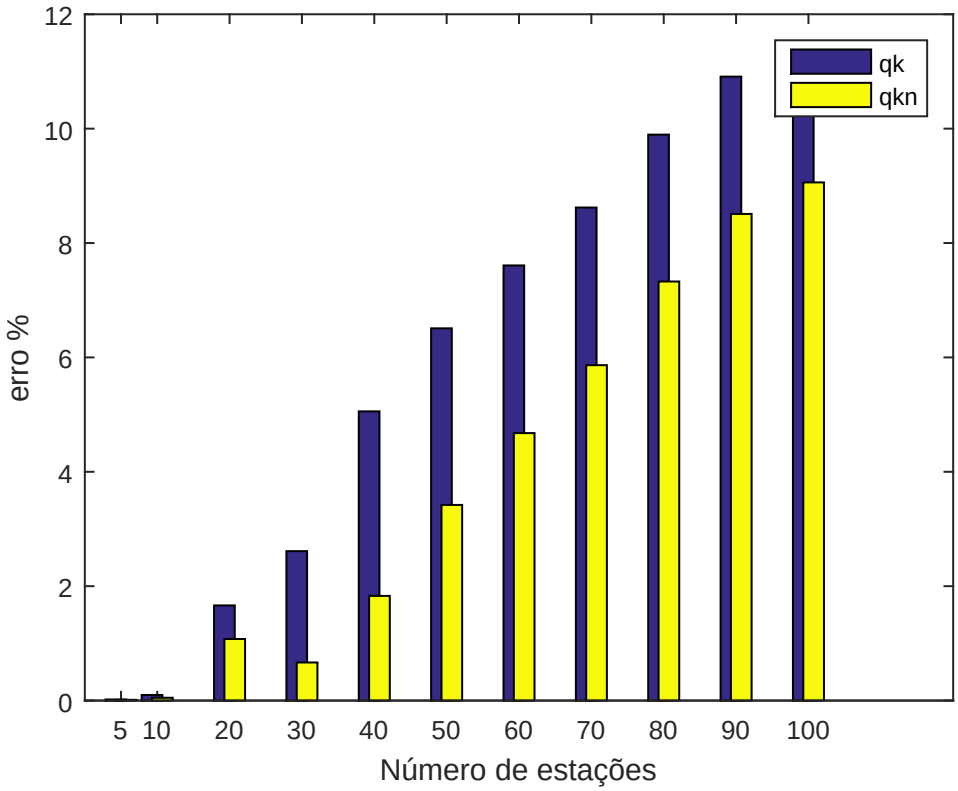

Figura 4.6: Gráfico de barras com erro percentual entre modelo analítico e simulação no NS3 para 10 slots.

lidade q_k proposta na Seção 3.4, conforme a Eq.(4.1), o modelo analítico fique mais próximo da simulação. Portanto,

$$q = c \times q_k, \tag{4.1}$$

em que q_k é a probabilidade de término do slot segundo a Eq.(3.9). Escolhemos a probabilidade q_k, que depende somente da probabilidade de recuo k, porque os resultados do modelo com as duas expressões empíricas foram semelhantes. Caso utilizássemos a probabilidade q_k^n na Eq.(4.1), mudaria apenas os valores de c para chegarmos aos mesmos resultados. Sendo assim, as Figuras 4.9, 4.10, 4.11 e 4.12 apresentam os valores de c para o casos de 2 slots, 5 slots, 10 slots e 15 slots RAW, respectivamente.

De acordo com as simulações computacionais, o valor de c que melhor se ajustou o modelo analítico para o caso de 2 slots RAW foi $c = 0,92$. Para o caso de 5 slots obtivemos $c = 0,75$; para o caso de 10 slots foi $c = 0,62$ e para o caso de 15 slots foi $c = 0,045$. As Figuras 4.9, 4.10, 4.11 e 4.12, contém os resultados para cada caso respectivo. Como podemos observar, o valor de c diminui com o aumento do número de slots RAW. Nota-se que, quando o número de slots é pequeno, por exemplo, quando há 2 slots RAW apenas, o valor de c é próximo de 1, o que indica que a expressão obtida para q_k está bem próxima do necessário, sendo necessário muito pouco ajuste. À medida que o número de slots RAW aumenta e o número de estações alocadas por slot

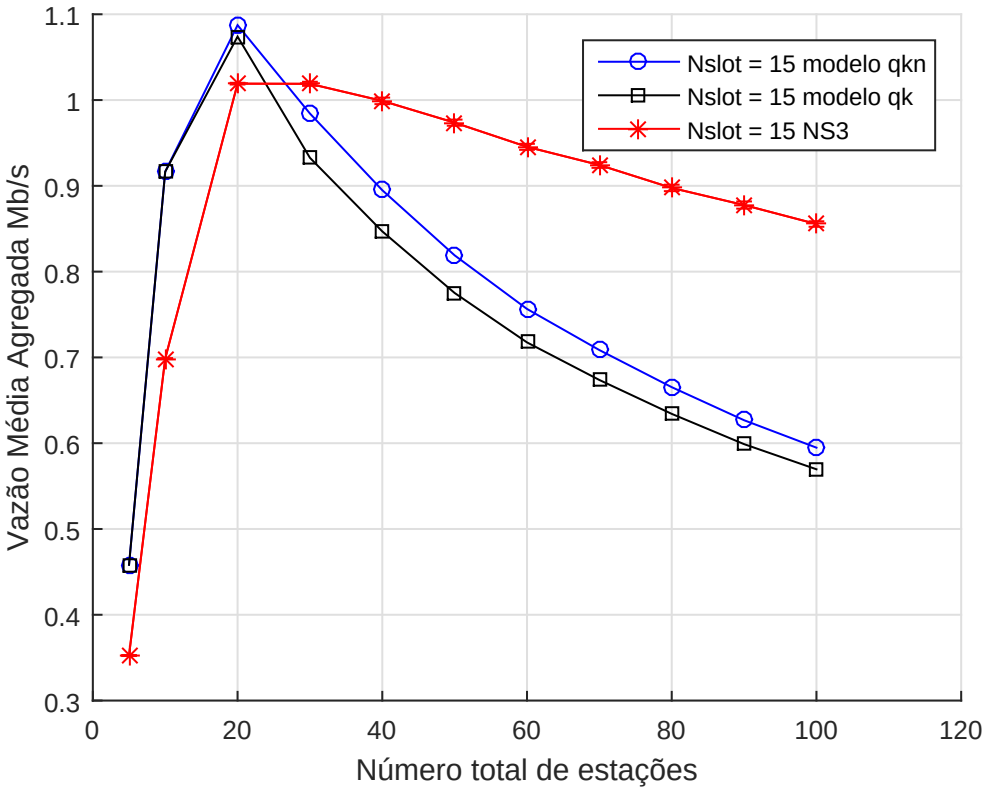

Figura 4.7: Comparação modelo analítico com simulação no NS3 com 15 slots RAW.

diminui (acompanhado da diminuição do tamanho de cada slot RAW), notamos que o fator de escala diminui progressivamente, distanciando-se de 1, mostrando que a aproximação empírica distancia-se do valor "necessário" cada vez mais. Como vimos, no caso em que temos 15 slots RAW, as discrepâncias foram maiores, e isto indica que a expressão empírica necessita ser melhor calculada para estes casos com menos estações e pequeno slot RAW.

Este capítulo apresentou os resultados da vazão média agregada da rede ao variar o número total de estações na rede para diferentes números de slots RAW por intervalo de *beacon*. Apresentamos os resultados considerando as duas abordagens para o cálculo da probabilidade de término de slot q e comparamos com os resultados obtidos por meio de simulações no simulador de redes NS3. Por fim, apresentamos os resultados que, ao multiplicar o modelo teórico por uma constante c, obtivemos resultados mais próximos aos resultados obtidos com as simulações. Assim, concluímos com este capítulo que a probabilidade de término de slot q tem grande importância no modelo matemático, pois ajustando-a, é possível obter curvas bem próximas das obtidas nas simulações. Desta forma, fica como trabalho futuro encontrar uma expressão empírica para q que melhor represente o comportamento da estação ao sair do estado ativo para o estado inativo, quando o seu slot RAW termina.

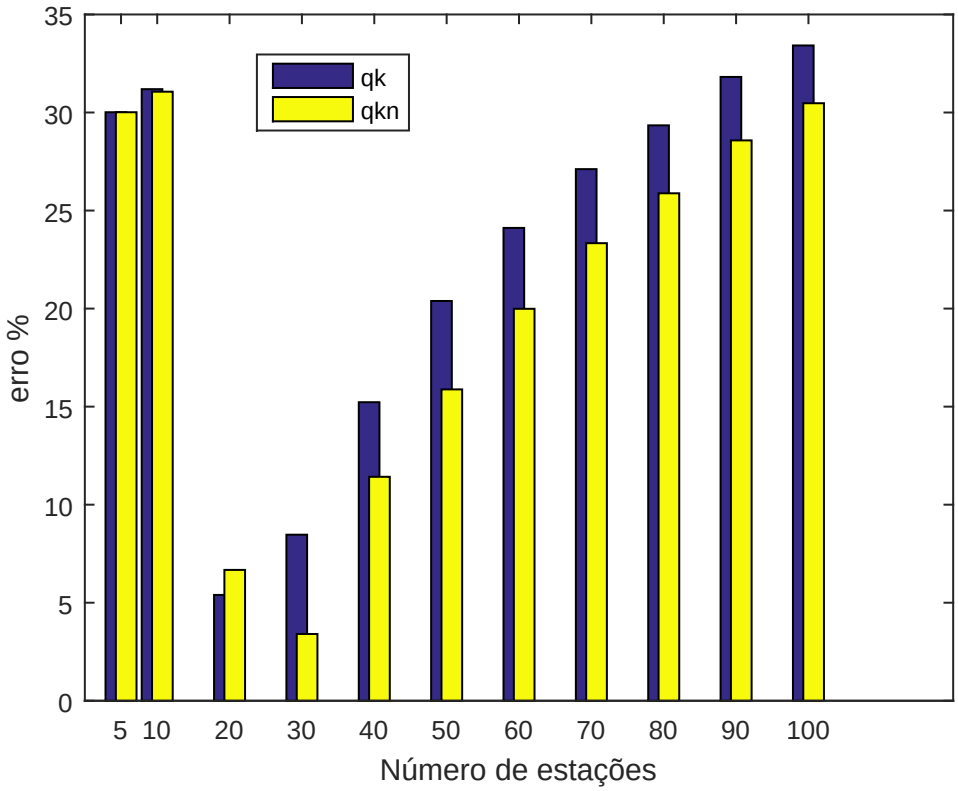

Figura 4.8: Gráfico de barras com erro percentual entre modelo analítico e simulação no NS3 para 15 slots.

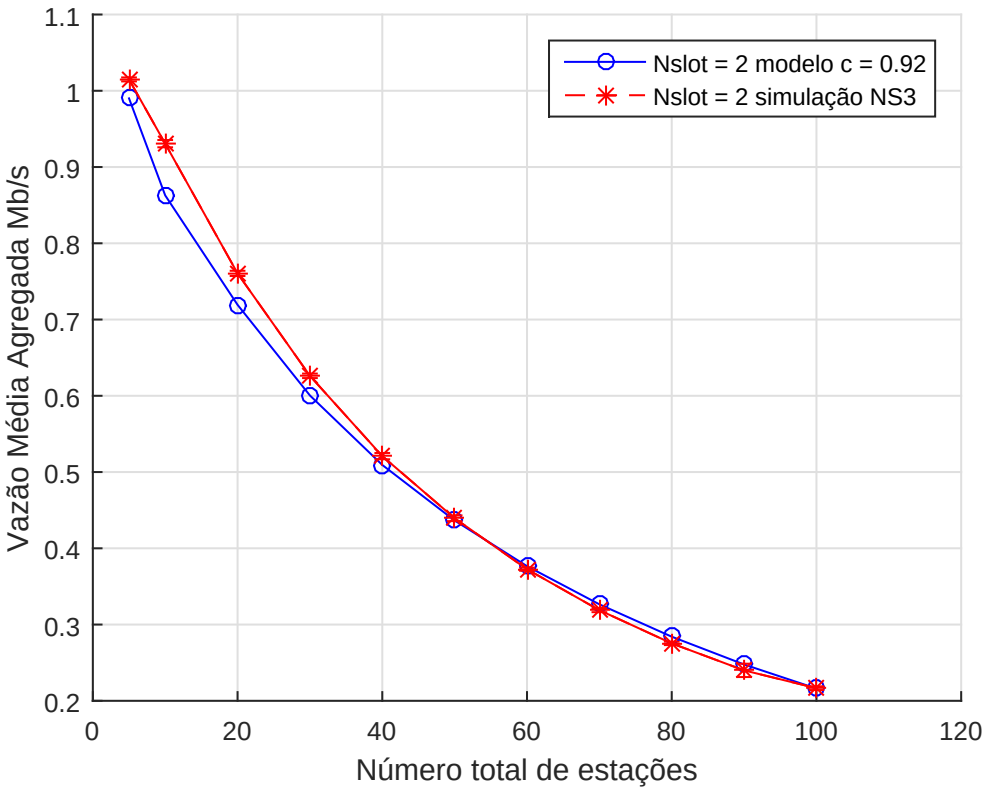

Figura 4.9: Comparação modelo analítico utilizando constante de ajuste $c = 0{,}92$ com simulação no NS3 para o caso com 2 slots RAW.

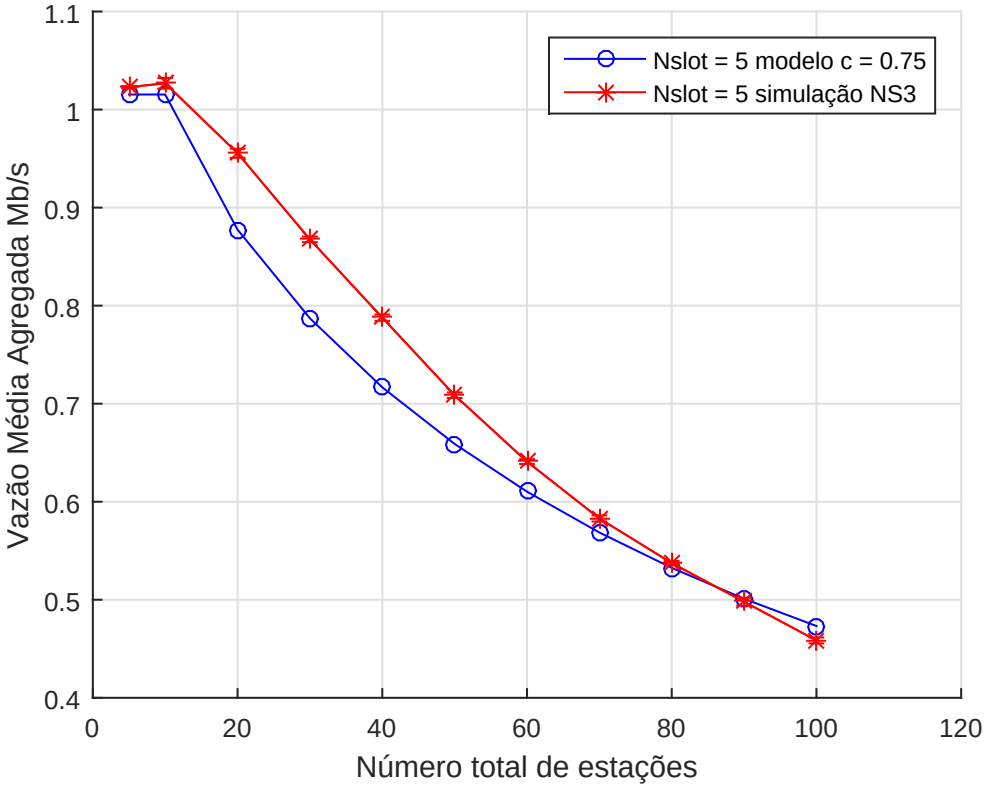

Figura 4.10: Comparação modelo analítico utilizando constante de ajuste $c = 0{,}75$ com simulação no NS3 para o caso com 5 slots RAW.

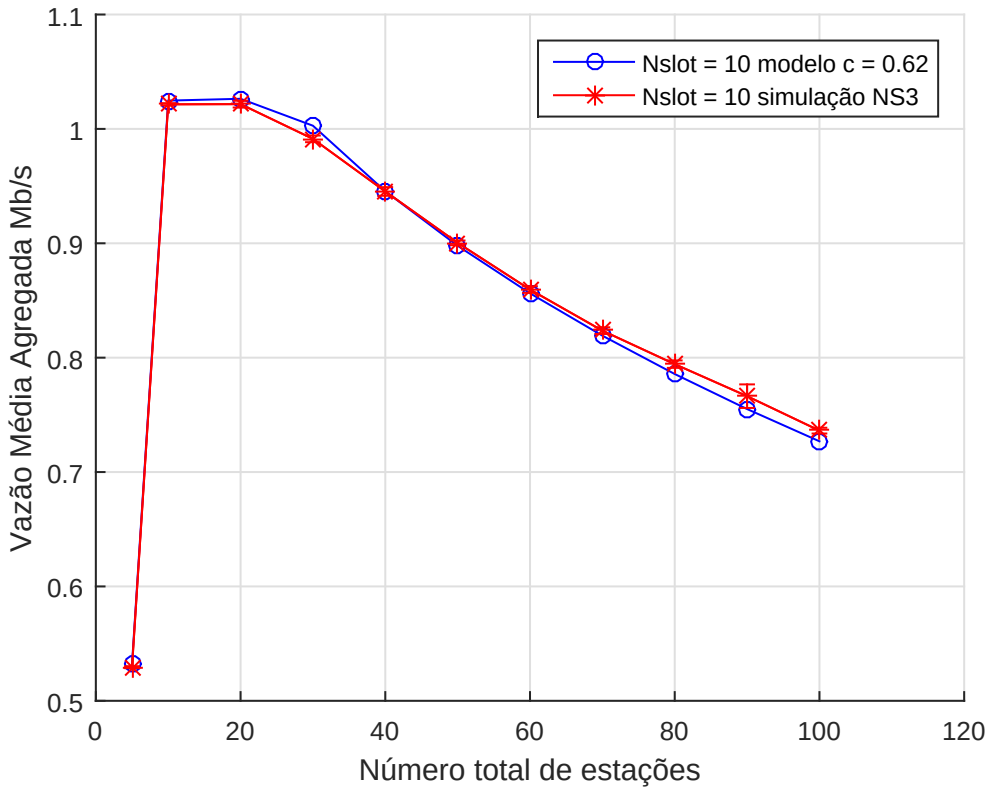

Figura 4.11: Comparação modelo analítico utilizando constante de ajuste $c = 0{,}62$ com simulação no NS3 para o caso com 10 slots RAW.

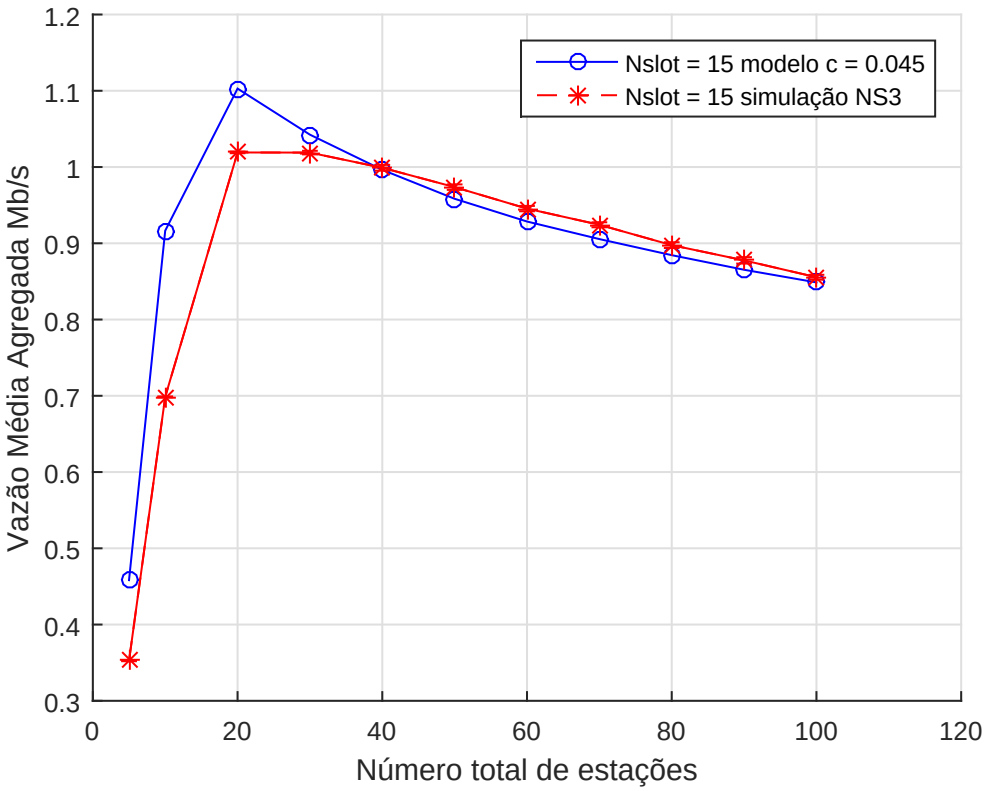

Figura 4.12: Comparação modelo analítico utilizando constante de ajuste $c = 0{,}045$ com simulação no NS3 para o caso com 15 slots RAW.

5 CONCLUSÕES

Atualmente, percebe-se o crescente uso de objetos inseridos no contexto da Internet das Coisas, cujos principais desafios são redes com muitos dispositivos conectados, com recursos de energia limitados e distribuídos geograficamente sobre áreas extensas. Para lidar com esses desafios, foi desenvolvido um novo padrão Wi-Fi, o IEEE 802.11ah, que consiste em comunicação de longo alcance que suporta muitos dispositivos conectados. Uma de suas principais características é o mecanismo de acesso restrito ao canal, o RAW. Com o mecanismo RAW, as estações da rede são divididas em grupos menores, que podem tentar o acesso ao canal somente no intervalo de tempo determinado para o grupo ao qual pertencem. Neste trabalho, foi proposto um modelo analítico do padrão IEEE 802.11ah para o cenário em que as estações têm tráfego saturado e condições ideais de canal, a fim de analisar o desempenho da vazão média agregada de uma rede ao utilizar o mecanismo de acesso restrito ao canal.

No mecanismo RAW, as estações realizam dois processos de *backoff*, o primeiro processo de *backoff* é utilizado fora do RAW e o segundo é utilizado dentro do slot RAW no qual a estação pode transmitir o seu pacote. As estações realizam o primeiro *backoff*, suspendendo-o assim que um grupo RAW inicia. As estações que não pertencem ao grupo RAW vão para o estado inativo e, assim, permanecem até o final do RAW, e as estações que pertencem ao grupo RAW iniciam o segundo processo de *backoff* durante o seu slot RAW atribuído, retornando ao estado inativo ao término do slot RAW. Quando o intervalo de tempo determinado ao RAW termina, todas as estações retornam ao estado do primeiro *bakoff* no qual parou. O objetivo deste trabalho foi desenvolver um modelo teórico que refletisse o comportamento de uma estação em atividade no seu slot RAW e sua transição entre o estado ativo e o estado inativo.

Primeiramente, foi apresentado o modelo analítico que consiste em uma cadeia de Markov em tempo discreto com duas dimensões para descrever o comportamento de uma estação em seu processo de *backoff* dentro de um slot RAW. Os estados da cadeia representam o contador de *backoff*, as linhas representam o estágio de retransmissão e as probabilidades de transição representam os eventos de decremento do contador de *backoff*, de congelamento do contador quando o canal está ocupado e de término do slot. Nossa principal contribuição foi propor um modelo utilizando cadeia de Markov que descreve a dinâmica de uma rede estação no IEEE 802.11ah, incluindo o momento de término do slot no qual a estação deve transitar do estado ativo para o estado inativo. Por este motivo, apresentamos duas propostas empíricas para a probabilidade de término do slot, o q da cadeia de Markov. A partir da solução numérica da cadeia, obtivemos as probabilidades necessárias para o cálculo da vazão média agregada da rede e analisamos o desempenho desta vazão utilizando as duas propostas empíricas da probabilidade q de término do slot.

Em seguida, foram realizadas simulações computacionais no simulador NS3 a fim de comparar e validar o modelo analítico. Nas simulações no NS3, assim como no modelo, variamos

o número de estações total da rede e o número de slots no grupo RAW e comparamos as duas propostas da probabilidade q de término do slot com as simulações. Com a probabilidade q de término do slot dependendo apenas do estágio de recuo, as curvas tiveram o comportamento semelhante às curvas obtidas com a simulação no NS3. Entretanto, houve discrepância quanto ao valor da vazão, principalmente para o caso de 15 slots RAW, em que a curva do modelo decresce mais rápido que a curva da simulação. Com a probabilidade q de término do slot dependendo do estágio de recuo e do número de estações contidas no slot RAW, o comportamento das curvas se manteve, e as discrepâncias quanto ao valor da vazão diminuíram, sendo que a discrepância para o caso de 15 slots RAW ainda foi alta. A cadeia de Markov é um modelo estatístico que descreve o comportamento de uma rede em regime estacionário, quando todas as estações já estão em atividade nos seus respectivos processos de *backoff*. Dado que a duração do RAW é fixa, a janela de atividades das estações fica mais curta ao dividir o RAW em mais slots, sendo mais difícil para as estações alcançarem o estado estacionário. Além disto, são poucas estações dentro de cada slot, o que torna este cenário muito parecido com um cenário TDMA, em que cada estação possui seu slot para transmitir sem disputar o acesso ao canal. Portanto, dinâmica do cenário com o RAW dividido em 15 slots pode não está sendo representada corretamente pelo modelo. Todavia, este cenário com poucas estações contidas em um slot não é um cenário esperado no contexto de Internet das Coisas. Na Internet das Coisas, espera-se um número elevado de estações na rede disputando pelo canal, portanto, o cenário em que o modelo falha não é um cenário típico esperado para a Internet das Coisas.

Por fim, baseados nos trabalhos [11] e [12] em que foi encontrada a probabilidade de término de janela a partir dos valores obtidos nas simulações computacionais, encontramos, também por meio dos resultados obtidos nas simulações, um valor da probabilidade q de término do slot para cada caso de número de slots RAW que aproximasse o modelo da simulação. Foi encontrada uma constante c que, ao multiplicar a probabilidade q proposta, que depende somente do estágio de recuo k (apresentada na Seção 3.4), obtivemos curvas bem próximas das simulações. Com isso, concluímos que a probabilidade de término do slot tem um papel fundamental para a acurácia do modelo, pois é possível obter resultados muito próximos às simulações com o valor adequado de q. Deste modo, citamos como um trabalho futuro, encontrar uma expressão empírica para a probabilidade q que represente de forma adequada o comportamento da estação ao sair do estado ativo para o estado inativo.

O desenvolvimento de modelos analíticos que descrevem protocolos de comunicação são importantes para avaliar as métricas de desempenho de uma rede. A partir do modelo teórico, é possível avaliar como a rede se comporta com diferentes condições iniciais e com variações de seus parâmetros, o que permite a implementação de protocolos de comunicação mais eficientes e com baixo consumo de energia. Assim, pode-se citar como trabalhos futuros, a expansão do modelo analítico para calcular outras métricas de desempenho, tais como o consumo de energia e o atraso de uma rede baseada no IEEE 802.11ah. Também seria interessante expandir o modelo para o caso de tráfego não saturado, dado que este é um cenário mais realista para a Internet das Coisas. Como já citado anteriormente, propor expressões empíricas para a probabilidade q de

término do slot que represente bem o comportamento da estação ao término do slot é um trabalho futuro muito importante, pois, como vimos neste trabalho, a probabilidade q influencia significativamente na acurácia do modelo. Pode-se citar também, a análise do modelo e de simulações com outros cenários como, por exemplo, variar o número de grupos RAW dentro de um intervalo de beacon. Além destes, ainda podemos citar como trabalhos futuros, propostas de novas formas de agrupamento das estações com análises realizadas a partir do modelo para aumentar a eficiência do IEEE 802.11ah e a realização de experimentos práticos de uma rede IEEE 802.11ah a fim de verificar a acurácia do modelo.

REFERÊNCIAS BIBLIOGRÁFICAS

1 TIAN, L.; DERONNE, S.; LATRÉ, S.; FAMAEY, J. Implementation and validation of an IEEE 802.11 ah module for ns-3. In: ACM. *Proceedings of the Workshop on ns-3*. [S.l.], 2016. p. 49–56.

2 ADAME, T.; BEL, A.; BELLALTA, B.; BARCELO, J.; OLIVER, M. IEEE 802.11 ah: the WiFi approach for M2M communications. *IEEE Wireless Communications*, IEEE, v. 21, n. 6, p. 144–152, 2014.

3 ERICSSON, A. Cellular networks for massive IoT—enabling low power wide area applications. *no. January*, p. 1–13, 2016.

4 AKELLA, A.; JUDD, G.; SESHAN, S.; STEENKISTE, P. Self-management in chaotic wireless deployments. *Wireless Networks*, Springer-Verlag New York, Inc., v. 13, n. 6, p. 737–755, 2007.

5 BELLALTI, B.; BONONI, L.; BRUNO, R.; KASSLER, A. Next generation IEEE 802.11 wireless local area networks: Current status, future directions and open challenges. *Computer Communications*, Elsevier, v. 75, p. 1–25, 2016.

6 RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low power wide area networks: An overview. *IEEE Communications Surveys & Tutorials*, IEEE, v. 19, n. 2, p. 855–873, 2017.

7 LIN, C. R.; GERLA, M. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected areas in Communications*, IEEE, v. 15, n. 7, p. 1265–1275, 1997.

8 HOU, T.-C.; TSAI, T.-J. A access-based clustering protocol for multihop wireless ad hoc networks. *IEEE journal on selected areas in communications*, IEEE, v. 19, n. 7, p. 1201–1210, 2001.

9 YU, J. Y.; CHONG, P. H. J. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, IEEE, v. 7, n. 1, p. 32–48, 2005.

10 SOARES, S. M.; CARVALHO, M. M. Revisiting the analytical modeling of the IEEE 802.11 power save mode for independent basic service sets (IBSS). In: ACM. *Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*. [S.l.], 2017. p. 17–24.

11 SWAIN, P.; CHAKRABORTY, S.; NANDI, S.; BHADURI, P. Performance modeling and evaluation of IEEE 802.11 IBSS power save mode. *Ad Hoc Networks*, Elsevier, v. 13, p. 336–350, 2014.

12 SWAIN, P.; CHAKRABORTY, S.; NANDI, S.; BHADURI, P. Performance modeling and analysis of IEEE 802.11 IBSS PSM in different traffic conditions. *IEEE Transactions on Mobile Computing*, IEEE, v. 14, n. 8, p. 1644–1658, 2015.

13 AHMED, N.; RAHMAN, H.; HUSSAIN, M. I. A comparison of 802.11 ah and 802.15. 4 for IoT. *ICT Express*, Elsevier, v. 2, n. 3, p. 100–102, 2016.

14 KHOROV, E.; LYAKHOV, A.; KROTOV, A.; GUSCHIN, A. A survey on IEEE 802.11 ah: An enabling networking technology for smart cities. *Computer Communications*, Elsevier, v. 58, p. 53–69, 2015.

15 BAÑOS-GONZALEZ, V.; AFAQUI, M. S.; LOPEZ-AGUILERA, E.; GARCIA-VILLEGAS, E. IEEE 802.11 ah: A technology to face the IoT challenge. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 16, n. 11, p. 1960, 2016.

16 LI, S.; XU, L. D.; ZHAO, S. The internet of things: a survey. *Information Systems Frontiers*, Springer, v. 17, n. 2, p. 243–259, 2014.

17 ZHENG, L.; NI, M.; CAI, L.; PAN, J.; GHOSH, C.; DOPPLER, K. Performance analysis of group-synchronized DCF for dense IEEE 802.11 networks. *IEEE Transactions on Wireless Communications*, IEEE, v. 13, n. 11, p. 6180–6192, 2014.

18 LIAO, R.; BELLALTA, B.; OLIVER, M.; NIU, Z. MU-MIMO MAC protocols for wireless local area networks: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 1, p. 162–183, 2016.

19 KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet (5ª edição). In: . [S.l.]: São Paulo, SP: Pearson Addison Wesley, 2010.

20 TOBAGI, F.; KLEINROCK, L. et al. Packet switching in radio channels: Part II–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *Communications, IEEE Transactions on*, IEEE, v. 23, n. 12, p. 1417–1433, 1975.

21 HAZMI, A.; RINNE, J.; VALKAMA, M. Feasibility study of IEEE 802.11 ah radio technology for IoT and M2M use cases. In: IEEE. *Globecom Workshops (GC Wkshps), 2012 IEEE*. [S.l.], 2012. p. 1687–1692.

22 KHOROV, E.; KROTOV, A.; LYAKHOV, A. Modelling machine type communication in IEEE 802.11 ah networks. In: IEEE. *Communication Workshop (ICCW), 2015 IEEE International Conference on*. [S.l.], 2015. p. 1149–1154.

23 RAEESI, O.; PIRSKANEN, J.; HAZMI, A.; LEVANEN, T.; VALKAMA, M. Performance evaluation of IEEE 802.11 ah and its restricted access window mechanism. In: IEEE. *Communications Workshops (ICC), 2014 EEE International Conference on*. [S.l.], 2014. p. 460–466.

24 PARK, C. W.; HWANG, D.; LEE, T.-J. Enhancement of IEEE 802.11 ah MAC for M2M communications. *IEEE Communications Letters*, IEEE, v. 18, n. 7, p. 1151–1154, 2014.

25 BIANCHI, G. Performance analysis of the IEEE 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, IEEE, v. 18, n. 3, p. 535–547, 2000.

26 ZHENG, L.; CAI, L.; PAN, J.; NI, M. Performance analysis of grouping strategy for dense IEEE 802.11 networks. In: IEEE. *Global Communications Conference (GLOBECOM), 2013 IEEE*. [S.l.], 2013. p. 219–224.

27 HAZMI, A.; BADIHI, B.; LARMO, A.; TORSNER, J.; VALKAMA, M. et al. Performance analysis of IoT-enabling IEEE 802.11 ah technology and its RAW mechanism with non-cross slot boundary holding schemes. In: IEEE. *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*. [S.l.], 2015. p. 1–6.

28 CHANG, T.-C.; LIN, C.-H.; LIN, K. C.-J.; CHEN, W.-T. Load-balanced sensor grouping for IEEE 802.11 ah networks. In: IEEE. *Global Communications Conference (GLOBECOM), 2015 IEEE*. [S.l.], 2015. p. 1–6.

29 DONG, M.; WU, Z.; GAO, X.; ZHAO, H. An efficient spatial group restricted access window scheme for IEEE 802.11 ah networks. In: IEEE. *Information Science and Technology (ICIST), 2016 Sixth International Conference on*. [S.l.], 2016. p. 168–173.

30 YOON, S.-G.; SEO, J.-O.; BAHK, S. Regrouping algorithm to alleviate the hidden node problem in 802.11 ah networks. *Computer Networks*, Elsevier, v. 105, p. 22–32, 2016.

APÊNDICES

I. SIMULAÇÕES NO NS3

Este script de simulação foi realizado na versão 3.23 do NS3 e o módulo do IEEE 802.11ah foi disponibilizado por Le tian et al. [1] em https://github.com/MOSAIC-UA/802.11ah-ns3. Segue abaixo, o único programa alterado para a realização de nossas simulações.

s1g-mac-test.cc

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2009 MIRKO BANCHI
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*/
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <ctime>
#include <fstream>
#include <sys/stat.h>

using namespace std;
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("s1g-wifi-network-le");

uint32_t AssocNum = 0;
int64_t AssocTime=0;
uint32_t StaNum=0;
string TrafficInterval="1000";
uint32_t payloadLength;

class assoc_record
{
public:
assoc_record ();
bool GetAssoc ();
void SetAssoc (std::string context, Mac48Address address);
void UnsetAssoc (std::string context, Mac48Address address);
void setstaid (uint16_t id);
private:
bool assoc;
uint16_t staid;
};

assoc_record::assoc_record ()
{
assoc = false;
staid = 65535;
}

void
assoc_record::setstaid (uint16_t id)
```

```
{
staid = id;
}

void
assoc_record::SetAssoc (std::string context, Mac48Address address)
{
assoc = true;
}

void
assoc_record::UnsetAssoc (std::string context, Mac48Address address)
{
assoc = false;
}

bool
assoc_record::GetAssoc ()
{
return assoc;
}

typedef std::vector <assoc_record * > assoc_recordVector;
assoc_recordVector assoc_vector;


uint32_t
GetAssocNum ( )
{
AssocNum = 0;
for (assoc_recordVector::const_iterator index = assoc_vector.begin ();
 index != assoc_vector.end (); index++)
{
if ((*index)->GetAssoc ())
{
AssocNum++;
}
}
return AssocNum;
}


void
PhyRxErrorTrace (std::string context, Ptr<const Packet> packet, double snr)
{
//  std::cout « "PHYRXERROR snr=" « snr « " " « *packet « std::endl;
}

void
PhyRxOkTrace (std::string context, Ptr<const Packet> packet, double snr,
WifiMode mode, enum WifiPreamble preamble)
{
//  std::cout « "PHYRXOK mode=" « mode « " snr=" « snr « " " « *packet « std::endl;
}


void
UdpTraffic ( string Interval, uint16_t num, uint32_t Size)
{
TrafficInterval = Interval;
StaNum = num;
payloadLength = Size;
}

ApplicationContainer serverApp;
typedef std::vector <Ptr<WifiMacQueue> > QueueVector;
QueueVector m_queue;
QueueVector BEqueue;

uint32_t NPacketInQueue = 0;
uint32_t AppStartTime = 0;
uint32_t ApStopTime = 0;

void ReadQueue(QueueVector queuelist)
{
BEqueue = queuelist;
}

uint16_t NEmptyQueue=0;
uint16_t kk=0;
```

```
void GetQueuelen (uint32_t Nsta)
{
NEmptyQueue=0;
kk=0;
for (QueueVector::const_iterator i = BEqueue.begin (); i != BEqueue.end (); i++)
{
kk +=1;
if ((*i)->GetSize () != 0)
{
// std::cout « kk «"th station, " « (*i)->GetSize () « " packets remains, "
« Simulator::Now ().GetSeconds () « std::endl;
// std::cout « "passou aqui leitura de fila"« std::endl;

}

if ((*i)->GetSize () == 0)
{
NEmptyQueue +=1 ;
}
}
///// comentei aqui //////////
if (NEmptyQueue == Nsta) // =1
//  if (NEmptyQueue == 1)
{
Simulator::Stop (Seconds (0));
//Simulator::Stop (Seconds (simulationTime+1));
//ApStopTime = Simulator::Now ().GetSeconds ();
//  std::cout « "no packets in the queue, "
« Simulator::Now ().GetSeconds () « std::endl;
std::cout « "Simulator start at " « AppStartTime « " s,
 end at " « ApStopTime « " s"  « std::endl;
//std::cout « "passou aqui "« std::endl;
}

else
{
Simulator::Schedule(Seconds (1.0), &GetQueuelen, Nsta);
// std::cout « "there are still packets in the queue" « std::endl;
}
/////////////// até aqui ////////////
}


void CheckAssoc (uint32_t Nsta, double simulationTime, NodeContainer wifiApNode,
NodeContainer  wifiStaNode, Ipv4InterfaceContainer apNodeInterface)
{
if  (GetAssocNum () == Nsta)
{
//std::cout « "Assoc Finished, AssocNum=" « AssocNum « ", time = "
 « Simulator::Now ().GetMicroSeconds () « std::endl;
//Application start time
double randomInterval = std::stod (TrafficInterval,nullptr);
Ptr<UniformRandomVariable> m_rv = CreateObject<UniformRandomVariable> ();
//UDP flow

UdpServerHelper myServer (9);
serverApp = myServer.Install (wifiApNode);
serverApp.Start (Seconds (0));
serverApp.Stop (Seconds (simulationTime+1)); // serverApp stops when simulator stop
// Set traffic start time for each station

UdpClientHelper myClient (apNodeInterface.GetAddress (0), 9); //address of remote node
myClient.SetAttribute ("MaxPackets", UintegerValue (4294967295u));
myClient.SetAttribute ("Interval", TimeValue (Time (TrafficInterval))); //packets/s
myClient.SetAttribute ("PacketSize", UintegerValue (payloadLength));
for (uint16_t i = 0; i < Nsta; i++)
{
double randomStart = m_rv->GetValue (0, randomInterval);
// std::cout « "Generate traffic in " « randomStart « "seconds, station " « double(i) « std::endl;

ApplicationContainer clientApp = myClient.Install (wifiStaNode.Get(i));
clientApp.Start (Seconds (1 + randomStart));
clientApp.Stop (Seconds (simulationTime+1));
// std::cout « "client stop at = " « Simulator::Now ().GetSeconds () « std::endl;
}
AppStartTime=Simulator::Now ().GetSeconds ();//+1; //clientApp start time
Simulator::Schedule(Seconds (simulationTime+1), &GetQueuelen, Nsta);
 //check before simulation stop       //estava esse
Simulator::Stop (Seconds (simulationTime+1)); //set stop time until no packet in queue
ApStopTime = Simulator::Now ().GetSeconds () + simulationTime;
std::cout « "Simulator start at " « AppStartTime « " s,
```

```
end at " « ApStopTime « " s"  « std::endl;

}
else
{
Simulator::Schedule(Seconds(1.0), &CheckAssoc, Nsta, simulationTime, wifiApNode,
 wifiStaNode, apNodeInterface);
}

}


void
PopulateArpCache ()
{
Ptr<ArpCache> arp = CreateObject<ArpCache> ();
arp->SetAliveTimeout (Seconds(3600 * 24 * 365));
for (NodeList::Iterator i = NodeList::Begin(); i != NodeList::End(); ++i)
{
Ptr<Ipv4L3Protocol> ip = (*i)->GetObject<Ipv4L3Protocol> ();
NS_ASSERT(ip !=0);
ObjectVectorValue interfaces;
ip->GetAttribute("InterfaceList", interfaces);
for(ObjectVectorValue::Iterator j = interfaces.Begin(); j !=
interfaces.End (); j ++)
{
Ptr<Ipv4Interface> ipIface = (j->second)->GetObject<Ipv4Interface> ();
NS_ASSERT(ipIface != 0);
Ptr<NetDevice> device = ipIface->GetDevice();
NS_ASSERT(device != 0);
Mac48Address addr = Mac48Address::ConvertFrom(device->GetAddress ());
for(uint32_t k = 0; k < ipIface->GetNAddresses (); k ++)
{
Ipv4Address ipAddr = ipIface->GetAddress (k).GetLocal();
if(ipAddr == Ipv4Address::GetLoopback())
continue;
ArpCache::Entry * entry = arp->Add(ipAddr);
entry->MarkWaitReply(0);
entry->MarkAlive(addr);
//   std::cout « "Arp Cache: Adding the pair (" « addr « "," « ipAddr « ")" « std::endl;
}
}
}
for (NodeList::Iterator i = NodeList::Begin(); i != NodeList::End(); ++i)
{
Ptr<Ipv4L3Protocol> ip = (*i)->GetObject<Ipv4L3Protocol> ();
NS_ASSERT(ip !=0);
ObjectVectorValue interfaces;
ip->GetAttribute("InterfaceList", interfaces);
for(ObjectVectorValue::Iterator j = interfaces.Begin(); j !=interfaces.End (); j ++)
{
Ptr<Ipv4Interface> ipIface = (j->second)->GetObject<Ipv4Interface> ();
ipIface->SetAttribute("ArpCache", PointerValue(arp));
}
}
}


int main (int argc, char *argv[])
{
//  LogComponentEnable ("UdpServer", LOG_INFO);
double simulationTime = 10;
uint32_t seed = 1;
uint32_t  payloadSize = 100;
uint32_t Nsta =1;
uint32_t NRawSta = 1;
uint32_t SlotFormat=1;
uint32_t  NRawSlotCount = 829;
uint32_t  NRawSlotNum = 1;
uint32_t NGroup = 1;
uint32_t BeaconInterval = 102400;
bool OutputPosition = true;
string DataMode = "OfdmRate2_4MbpsBW1MHz";
double datarate = 2.4;
double bandWidth = 1;
string UdpInterval="0.2";
string rho="250.0";
string folder="./scratch/";
string file="./scratch/mac-sta.txt";
string pcapfile="./scratch/mac-slg-slots";
```

```
CommandLine cmd;
cmd.AddValue ("seed", "random seed", seed);
cmd.AddValue ("simulationTime", "Simulation time in seconds", simulationTime);
cmd.AddValue ("payloadSize", "Size of payload", payloadSize);
cmd.AddValue ("Nsta", "number of total stations", Nsta);
cmd.AddValue ("NRawSta", "number of stations supporting RAW", NRawSta);
cmd.AddValue ("SlotFormat", "format of NRawSlotCount", SlotFormat);
cmd.AddValue ("NRawSlotCount", "number of stations supporting RAW", NRawSlotCount);
cmd.AddValue ("NRawSlotNum", "number of stations supporting RAW", NRawSlotNum);
cmd.AddValue ("NGroup", "number of RAW group", NGroup);
cmd.AddValue ("BeaconInterval", "Beacon interval time in us", BeaconInterval);
cmd.AddValue ("DataMode", "Date mode", DataMode);
cmd.AddValue ("datarate", "data rate in Mbps", datarate);
cmd.AddValue ("bandWidth", "bandwidth in MHz", bandWidth);
cmd.AddValue ("UdpInterval", "traffic mode", UdpInterval);
cmd.AddValue ("rho", "maximal distance between AP and stations", rho);
cmd.AddValue ("folder", "folder where result files are placed", folder);
cmd.AddValue ("file", "files containing reslut information", file);
cmd.AddValue ("pcapfile", "files containing reslut information", pcapfile);
cmd.Parse (argc,argv);

RngSeedManager::SetSeed (seed);

NodeContainer wifiStaNode;
wifiStaNode.Create (Nsta);
NodeContainer wifiApNode;
wifiApNode.Create (1);

YansWifiChannelHelper channel = YansWifiChannelHelper ();
//channel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel","Exponent",
 DoubleValue(3.67) ,"ReferenceLoss", DoubleValue(8), "ReferenceDistance",
  DoubleValue(1.0), "Frequency", DoubleValue(868e6));
channel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue (-80.0));
channel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");

YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetErrorRateModel ("ns3::YansErrorRateModel");
phy.SetChannel (channel.Create ());
phy.Set ("ShortGuardEnabled", BooleanValue (false));
phy.Set ("ChannelWidth", UintegerValue (bandWidth));
phy.Set ("EnergyDetectionThreshold", DoubleValue (-116.0));
phy.Set ("CcaMode1Threshold", DoubleValue (-119.0));
phy.Set ("FadingMargin", DoubleValue (0));
phy.Set ("TxGain", DoubleValue (0.0));
phy.Set ("RxGain", DoubleValue (0.0));
phy.Set ("TxPowerLevels", UintegerValue (1));
phy.Set ("TxPowerEnd", DoubleValue (0.0));
phy.Set ("TxPowerStart", DoubleValue (0.0));
phy.Set ("RxNoiseFigure", DoubleValue (3.0));
//phy.Set ("RxNoiseFigure", DoubleValue (0.0));
phy.Set ("LdpcEnabled", BooleanValue (true));

WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211ah);
S1gWifiMacHelper mac = S1gWifiMacHelper::Default ();

Ssid ssid = Ssid ("ns380211ah");
StringValue DataRate;
DataRate = StringValue (DataMode);

wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode", DataRate,
"ControlMode", DataRate);
mac.SetType ("ns3::StaWifiMac",
"Ssid", SsidValue (ssid),
"ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevice;
staDevice = wifi.Install (phy, mac, wifiStaNode);


Config::Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Mac/Slot",
TimeValue (MicroSeconds (52)));//tirar
Config::Set ("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Mac/Sifs",
 TimeValue (MicroSeconds (160)));//tirar


uint32_t NGroupStas = NRawSta/NGroup;
mac.SetType ("ns3::ApWifiMac",
"Ssid", SsidValue (ssid),
"BeaconInterval", TimeValue (MicroSeconds(BeaconInterval)),
```

```
"NRawGroupStas", UintegerValue (NGroupStas),
"NRawStations", UintegerValue (NRawSta),
"SlotFormat", UintegerValue (SlotFormat),
"SlotCrossBoundary", UintegerValue (0),
"SlotDurationCount", UintegerValue (NRawSlotCount),
"SlotNum", UintegerValue (NRawSlotNum));

NetDeviceContainer apDevice;
phy.Set ("FadingMargin", DoubleValue (0));
phy.Set ("TxGain", DoubleValue (3.0));
phy.Set ("RxGain", DoubleValue (3.0));
phy.Set ("TxPowerLevels", UintegerValue (1));
phy.Set ("TxPowerEnd", DoubleValue (30.0));
phy.Set ("TxPowerStart", DoubleValue (30.0));
phy.Set ("RxNoiseFigure", DoubleValue (5));
apDevice = wifi.Install (phy, mac, wifiApNode);

Config::Set ("/NodeList/*/DeviceList/0/$ns3::WifiNetDevice/Mac/$ns3::
RegularWifiMac/BE_EdcaTxopN/Queue/MaxPacketNumber", UintegerValue(60000));
Config::Set ("/NodeList/*/DeviceList/0/$ns3::WifiNetDevice/Mac/$ns3::
RegularWifiMac/BE_EdcaTxopN/Queue/MaxDelay", TimeValue (NanoSeconds (6000000000000)));

// mobility.
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::UniformDiscPositionAllocator",
"X", StringValue ("1000.0"),
"Y", StringValue ("1000.0"),
"rho", StringValue (rho));
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiStaNode);

MobilityHelper mobilityAp;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (1000.0, 1000.0, 0.0));
mobilityAp.SetPositionAllocator (positionAlloc);
mobilityAp.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobilityAp.Install(wifiApNode);


/* Internet stack*/
InternetStackHelper stack;
stack.Install (wifiApNode);
stack.Install (wifiStaNode);

Ipv4AddressHelper address;

address.SetBase ("192.168.0.0", "255.255.0.0");
Ipv4InterfaceContainer staNodeInterface;
Ipv4InterfaceContainer apNodeInterface;

staNodeInterface = address.Assign (staDevice);
apNodeInterface = address.Assign (apDevice);

//trace association
for (uint16_t kk=0; kk< Nsta; kk++)
{
std::ostringstream STA;
STA « kk;
std::string strSTA = STA.str();

assoc_record *m_assocrecord=new assoc_record;
m_assocrecord->setstaid (kk);
Config::Connect ("/NodeList/"+strSTA+"/DeviceList/0/$ns3::WifiNetDevice/Mac/
$ns3::RegularWifiMac/$ns3::StaWifiMac/Assoc",
 MakeCallback (&assoc_record::SetAssoc, m_assocrecord));
Config::Connect ("/NodeList/"+strSTA+"/DeviceList/0/$ns3::WifiNetDevice/Mac/
$ns3::RegularWifiMac/$ns3::StaWifiMac/DeAssoc",
 MakeCallback (&assoc_record::UnsetAssoc, m_assocrecord));
assoc_vector.push_back (m_assocrecord);
}


//trafego udp eu acrescentei, não é original

//Application start time
//  double randomInterval = std::stod (TrafficInterval,nullptr);
//Ptr<UniformRandomVariable> m_rv = CreateObject<UniformRandomVariable> ();
//UDP flow

// UdpServerHelper myServer (9);
// serverApp = myServer.Install (wifiApNode);
```

```
// serverApp.Start (Seconds (0));
// serverApp.Stop (Seconds (simulationTime+1)); // serverApp stops when simulator stop
//Set traffic start time for each station

//UdpClientHelper myClient (apNodeInterface.GetAddress (0), 9); //address of remote node
//myClient.SetAttribute ("MaxPackets", UintegerValue (4294967295));
// myClient.SetAttribute ("Interval", TimeValue (Time (TrafficInterval))); //packets/s
// myClient.SetAttribute ("PacketSize", UintegerValue (payloadLength));
// for (uint16_t i = 0; i < Nsta; i++)
//   {
//     double randomStart = m_rv->GetValue (0, randomInterval);
//     std::cout « "Generate traffic in " « randomStart « "seconds, station " « double(i) « std::endl;

//   ApplicationContainer clientApp =
myClient.Install (wifiStaNode.Get(i));
// clientApp.Start (Seconds (1 + randomStart));
// clientApp.Stop (Seconds (simulationTime+1));
// }
//  AppStartTime=Simulator::Now ().GetSeconds () + 1; //clientApp start time
// Simulator::Schedule(Seconds (simulationTime+1),
 &GetQueuelen, Nsta); //check before simulation stop       //estava esse
//  Simulator::Stop (Seconds (simulationTime+1));
//set stop time until no packet in queue

// check BE_EdcaTxopN queue length of stations
//********* talvez comentar essa parte ************************//
for (uint16_t i = 0; i < Nsta; i++)
{
Ptr<NetDevice> netDe;
Ptr<WifiNetDevice> wifiDe;
netDe = staDevice.Get(i);
wifiDe = netDe->GetObject<WifiNetDevice>();
PointerValue ptr1;
wifiDe->GetMac()->GetAttribute("BE_EdcaTxopN", ptr1);
Ptr<EdcaTxopN> BE_edca = ptr1.Get<EdcaTxopN>();
PointerValue ptr2;
BE_edca->GetAttribute("Queue", ptr2);
Ptr<WifiMacQueue> BE_edca_queue = ptr2.Get<WifiMacQueue> ();
m_queue.push_back (BE_edca_queue);
}

Simulator::ScheduleNow(&UdpTraffic, UdpInterval, Nsta,payloadSize);
Simulator::Schedule(Seconds(1), &ReadQueue, m_queue);
Simulator::Schedule(Seconds(1), &CheckAssoc, Nsta, simulationTime,
wifiApNode, wifiStaNode,apNodeInterface);
// ************************************************************//

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
PopulateArpCache ();

if  (OutputPosition)
{
int i =0;
while (i < Nsta)
{
Ptr<MobilityModel> mobility = wifiStaNode.Get (i)->GetObject<MobilityModel>();
Vector position = mobility->GetPosition();
//   std::cout « "Sta node#" « i « ", " « "position = " « position « std::endl;
i++;
}
Ptr<MobilityModel> mobility1 = wifiApNode.Get (0)->GetObject<MobilityModel>();
Vector position = mobility1->GetPosition();
//std::cout « "AP node, position = " « position « std::endl;
}

phy.EnablePcap ("pcapfile", Nsta, 0);
Simulator::Run ();
Simulator::Destroy ();


double throughput = 0;
//UDP
uint32_t totalPacketsThrough = DynamicCast<UdpServer> (serverApp.Get (0))->GetReceived ();
throughput = totalPacketsThrough * payloadSize * 8 / ((ApStopTime - AppStartTime) * 1000000.0);


// std::cout « "DataRate" « "\t" « "NGroup"
« "\t" « "NRawSlotNum" « "\t" « "Throughput" « '\n';
std::cout « "DataRate" « "\t" « "NGroup"
« "\t" « "NRawSlotNum" « "\t" « "Nsta"« "\t"« "Throughput" « '\n';
// std::cout « datarate « "\t" « NGroup « "\t"
```

```
« NRawSlotNum « "\t" « throughput « " Mbit/s" « std::endl;
std::cout « datarate « "\t" « NGroup « "\t" « NRawSlotNum
 « "\t" « Nsta « "\t" « throughput « " Mbit/s" « std::endl;
ofstream myfile;
myfile.open (file, ios::out | ios::app);
myfile « NGroup « "\t" « NRawSlotNum « "\t" « throughput «"\n";
myfile.close();

return 0;
}
```

I. PROGRAMAS NO MATLAB

Estes foram os programas para a solução numérica e o cálculo da vazão do modelo analítico.

```
% comparacão entre o modelo analítico e a simulaçao no NS3
% heuristicas qk e qkn
% 2 slots RAW

% solução de tau-d e estados bi,j
clc
% Definição de variáveis
basic_rate = 1000000; %bits/s
data_rate = 7800000; % bits/s
phy_header = 0.000192; %em segundos, parametro do ah
%h = 416/basic_rate; % (416 = 224 bits cabeçalho MAC + 192 bits cabeçalho
%PHY)em segundos, cabeçalho antigo
%ack = 304/basic_rate; % (304= 112 bits + 192 bits phy) em segundos
mac_header = (34*8)/basic_rate;
h = phy_header + mac_header;
ack = (14*8)/basic_rate + phy_header;
e = 256*8; %tamanho do payload/bitrate
e_duracao = 256*8/data_rate;
%sifs = 0.000160; % em segundos
%difs = 0.000303; % em segundos
%delta = 0.000006; % em segundos, estava 0.000001
%sigma = 0.000052; % em segundos
%ack_timeout = 0.000600;
sifs = 0.000160; % em segundos
difs = 0.000304; % em segundos
delta = 0.0000033; % em segundos, estava 0.000001
sigma = 0.000052; % em segundos
ack_timeout = 2*delta + sifs + ack;
beacon_interval = 0.1; %em segundos
%data_window_size = beacon_interval-20;
x0 = [0.01;0.01]; % valor inicial para fsolve
% tempo médio em que o canal está ocupado com uma transmissão com sucesso
Ts = h + e_duracao + sifs + (2.*delta) + ack + difs;
% tempo médio em que o canal está ocupado com uma trasmissão com colisão
Tc = h + e_duracao + difs + delta + ack_timeout;
% tempo de holding
Tg = 0.000008;
Th = Ts + 2*Tg;

Tidle_2 = (beacon_interval/2)-Th;
Tidle_5 = (beacon_interval/5);
Tidle_10 = (beacon_interval/10);
Tidle_15 = (beacon_interval/15);

simulacaons3_2slots;


% para Nslot = 2 (2 slots dentro do raw)

% para n = 5
fun1_2slots = @solucao_ah_v2_qvariavel_n5;
result1_2slots = fsolve(fun1_2slots,x0);
taud1_2slots = result1_2slots(1)
b0_01_2slots = result1_2slots(2)
n1 = 5;
n1_2slots = n1/2;

% cálculo das probabilidades
ptr1_2slots = 1 - ((1-taud1_2slots)^n1_2slots);
pds1_2slots = (n1_2slots*taud1_2slots*((1-taud1_2slots)^(n1_2slots-1)))/ptr1_2slots;


% Cálculo das probabilidades de transmissão e de sucess
t_slot1_2slots = ((1 - ptr1_2slots).*sigma +
 ptr1_2slots.*pds1_2slots.*Ts + ptr1_2slots.*(1 - pds1_2slots).*Tc);
s1_2slots = pds1_2slots.*ptr1_2slots.*e./t_slot1_2slots;
b_idle1 = 7.0809e-04;

% para n = 10
fun2_2slots = @solucao_ah_v2_qvariavel_n10;
result2_2slots = fsolve(fun2_2slots,x0)
taud2_2slots = result2_2slots(1);
```

```
b0_02_2slots =result2_2slots(2);
n2 =10;
n2_2slots = n2/2;


ptr2_2slots = 1 - ((1-taud2_2slots)^n2_2slots);
pds2_2slots = (n2_2slots*taud2_2slots*((1-taud2_2slots)^(n2_2slots-1)))/ptr2_2slots;
t_slot2_2slots = ((1 - ptr2_2slots).*sigma +
ptr2_2slots.*pds2_2slots.*Ts + ptr2_2slots.*(1 - pds2_2slots).*Tc);
s2_2slots = pds2_2slots.*ptr2_2slots.*e./t_slot2_2slots;

% para n = 20
fun3_2slots = @solucao_ah_v2_qvariavel_n20;
result3_2slots = fsolve(fun3_2slots,x0);
taud3_2slots = result3_2slots(1)
b0_03_2slots = result3_2slots(2)
n3 =20;
n3_2slots = n3/2;

ptr3_2slots = 1 - ((1-taud3_2slots)^n3_2slots);
pds3_2slots = (n3_2slots*taud3_2slots*((1-taud3_2slots)^(n3_2slots-1)))/ptr3_2slots;
t_slot3_2slots = ((1 - ptr3_2slots).*sigma +
 ptr3_2slots.*pds3_2slots.*Ts + ptr3_2slots.*(1 - pds3_2slots).*Tc);
s3_2slots = pds3_2slots.*ptr3_2slots.*e./t_slot3_2slots;

% para n = 30
fun4_2slots = @solucao_ah_v2_qvariavel_n30;
result4_2slots = fsolve(fun4_2slots,x0);
taud4_2slots = result4_2slots(1)
b0_04_2slots = result4_2slots(2)
n4 =30;
n4_2slots = n4/2;

ptr4_2slots = 1 - ((1-taud4_2slots)^n4_2slots);
pds4_2slots = (n4_2slots*taud4_2slots*((1-taud4_2slots)^(n4_2slots-1)))/ptr4_2slots;
t_slot4_2slots = ((1 - ptr4_2slots).*sigma +
 ptr4_2slots.*pds4_2slots.*Ts + ptr4_2slots.*(1 - pds4_2slots).*Tc);
s4_2slots = pds4_2slots.*ptr4_2slots.*e./t_slot4_2slots;

% para n = 40
fun5_2slots = @solucao_ah_v2_qvariavel_n40;
result5_2slots = fsolve(fun5_2slots,x0);
taud5_2slots = result5_2slots(1)
b0_05_2slots = result5_2slots(2)
n5 =40;
n5_2slots = n5/2;


ptr5_2slots = 1 - ((1-taud5_2slots)^n5_2slots);
pds5_2slots = (n5_2slots*taud5_2slots*((1-taud5_2slots)^(n5_2slots-1)))/ptr5_2slots;
t_slot5_2slots = ((1 - ptr5_2slots).*sigma +
 ptr5_2slots.*pds5_2slots.*Ts + ptr5_2slots.*(1 - pds5_2slots).*Tc);
s5_2slots = pds5_2slots.*ptr5_2slots.*e./t_slot5_2slots;


% para n = 50
fun6_2slots = @solucao_ah_v2_qvariavel_n50;
result6_2slots = fsolve(fun6_2slots,x0);
taud6_2slots = result6_2slots(1)
b0_06_2slots = result6_2slots(2)
n6 =50;
n6_2slots = n6/2;

ptr6_2slots = 1 - ((1-taud6_2slots)^n6_2slots);
pds6_2slots = (n6_2slots*taud6_2slots*((1-taud6_2slots)^(n6_2slots-1)))/ptr6_2slots;
t_slot6_2slots = ((1 - ptr6_2slots).*sigma +
 ptr6_2slots.*pds6_2slots.*Ts + ptr6_2slots.*(1 - pds6_2slots).*Tc);
s6_2slots = pds6_2slots.*ptr6_2slots.*e./t_slot6_2slots;


% para n = 60
fun7_2slots = @solucao_ah_v2_qvariavel_n60;
result7_2slots = fsolve(fun7_2slots,x0);
taud7_2slots = result7_2slots(1)
b0_07_2slots = result7_2slots(2)
n7 =60;
n7_2slots = n7/2;

ptr7_2slots = 1 - ((1-taud7_2slots)^n7_2slots);
pds7_2slots = (n7_2slots*taud7_2slots*((1-taud7_2slots)^(n7_2slots-1)))/ptr7_2slots;
t_slot7_2slots =((1 - ptr7_2slots).*sigma +
```

```
 ptr7_2slots.*pds7_2slots.*Ts + ptr7_2slots.*(1 - pds7_2slots).*Tc);
s7_2slots = pds7_2slots.*ptr7_2slots.*e./t_slot7_2slots;

% para n = 70
fun8_2slots = @solucao_ah_v2_qvariavel_n70;
result8_2slots = fsolve(fun8_2slots,x0);
taud8_2slots =result8_2slots(1)
b0_08_2slots = result8_2slots(2)
n8 =70;
n8_2slots = n8/2;

ptr8_2slots = 1 - ((1-taud8_2slots)^n8_2slots);
pds8_2slots = (n8_2slots*taud8_2slots*((1-taud8_2slots)^(n8_2slots-1)))/ptr8_2slots;
t_slot8_2slots = ((1 - ptr8_2slots).*sigma +
 ptr8_2slots.*pds8_2slots.*Ts + ptr8_2slots.*(1 - pds8_2slots).*Tc);
s8_2slots = pds8_2slots.*ptr8_2slots.*e./t_slot8_2slots;



% para n = 80
fun9_2slots = @solucao_ah_v2_qvariavel_n80;
result9_2slots = fsolve(fun9_2slots,x0);
taud9_2slots = result9_2slots(1)
b0_09_2slots = result9_2slots(2)
n9 = 80;
n9_2slots = n9/2;

% cálculo das probabilidades
ptr9_2slots = 1 - ((1-taud9_2slots)^n9_2slots);
pds9_2slots = (n9_2slots*taud9_2slots*((1-taud9_2slots)^(n9_2slots-1)))/ptr9_2slots;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9_2slots =((1 - ptr9_2slots).*sigma +
 ptr9_2slots.*pds9_2slots.*Ts + ptr9_2slots.*(1 - pds9_2slots).*Tc);
s9_2slots = pds9_2slots.*ptr9_2slots.*e./t_slot9_2slots;

% 90 estações
fun10_2slots = @solucao_ah_v2_qvariavel_n90;
result10_2slots = fsolve(fun10_2slots,x0);
taud10_2slots = result10_2slots(1)
b0_010_2slots = result10_2slots(2)
n10 =90;
n10_2slots = n10/2;

ptr10_2slots = 1 - ((1-taud10_2slots)^n10_2slots);
pds10_2slots = (n10_2slots*taud10_2slots*((1-taud10_2slots)^(n10_2slots-1)))/ptr10_2slots;
t_slot10_2slots = ((1 - ptr10_2slots).*sigma +
 ptr10_2slots.*pds10_2slots.*Ts + ptr10_2slots.*(1 - pds10_2slots).*Tc);
s10_2slots = pds10_2slots.*ptr10_2slots.*e./t_slot10_2slots;

% para n=100
fun11_2slots = @solucao_ah_v2_qvariavel_n100;
result11_2slots = fsolve(fun11_2slots,x0);
taud11_2slots = result11_2slots(1)
b0_011_2slots = result11_2slots(2)
n11 =100;
n11_2slots = 100/2;

ptr11_2slots = 1 - ((1-taud11_2slots)^n11_2slots);
pds11_2slots = (n11_2slots*taud11_2slots*((1-taud11_2slots)^(n11_2slots-1)))/ptr11_2slots;
t_slot11_2slots = ((1 - ptr11_2slots).*sigma +
 ptr11_2slots.*pds11_2slots.*Ts + ptr11_2slots.*(1 - pds11_2slots).*Tc)
s11_2slots = (pds11_2slots.*ptr11_2slots.*e)./t_slot11_2slots;


n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11];
vazao1_qkn =(((Tidle_2)/beacon_interval)/1000000).*[s1_2slots,s2_2slots,s3_2slots,s4_2slots,
s5_2slots,s6_2slots,s7_2slots,s8_2slots,s9_2slots,s10_2slots,s11_2slots].*2;
nrawsta=[5,10,20,30,40,50,60,70,80,90,100];

%vazão qk
% para Nslot = 10 (10 slots dentro do raw)
% para n = 5
fun1 = @solucao_ah_v2_qk_n5;
result1 = fsolve(fun1,x0);
taud1 = result1(1);
b0_01 = result1(2);
n1 = 5;

% cálculo das probabilidades
ptr1 = 1 - ((1-taud1)^n1_2slots);
pds1 = (n1_2slots*taud1*((1-taud1)^(n1_2slots-1)))/ptr1;
```

```
% Cálculo das probabilidades de transmissão e de sucess
t_slot1 = ((1 - ptr1).*sigma + ptr1.*pds1.*Ts + ptr1.*(1 - pds1).*Tc);
s1 = (pds1.*ptr1.*e./t_slot1);

% para n = 10
fun2 = @solucao_ah_v2_qk_n10;
result2 = fsolve(fun2,x0)
taud2 = result2(1);
b0_02 =result2(2);

ptr2 = 1 - ((1-taud2)^n2_2slots);
pds2 = (n2_2slots*taud2*((1-taud2)^(n2_2slots-1)))/ptr2;
t_slot2 = ((1 - ptr2).*sigma + ptr2.*pds2.*Ts + ptr2.*(1 - pds2).*Tc);
s2 = (pds2.*ptr2.*e./t_slot2);

% para n = 20
fun3 = @solucao_ah_v2_qk_n20;
result3 = fsolve(fun3,x0);
taud3 = result3(1)
b0_03 = result3(2)
n3 =20;

ptr3 = 1 - ((1-taud3)^n3_2slots);
pds3 = (n3_2slots*taud3*((1-taud3)^(n3_2slots-1)))/ptr3;
t_slot3 = ((1 - ptr3).*sigma + ptr3.*pds3.*Ts + ptr3.*(1 - pds3).*Tc);
s3 = pds3.*ptr3.*e./t_slot3;

% para n = 30
fun4 = @solucao_ah_v2_qk_n30;
result4 = fsolve(fun4,x0);
taud4 = result4(1);
b0_04 = result4(2);
n4 =30;

ptr4 = 1 - ((1-taud4)^n4_2slots);
pds4 = (n4_2slots*taud4*((1-taud4)^(n4_2slots-1)))/ptr4;
t_slot4 = ((1 - ptr4).*sigma + ptr4.*pds4.*Ts + ptr4.*(1 - pds4).*Tc);
s4 = pds4.*ptr4.*e./t_slot4;

% para n = 40
fun5 = @solucao_ah_v2_qk_n40;
result5 = fsolve(fun5,x0);
taud5 = result5(1)
b0_05 = result5(2)
n5 =40;

ptr5 = 1 - ((1-taud5)^n5_2slots);
pds5 = (n5_2slots*taud5*((1-taud5)^(n5_2slots-1)))/ptr5;
t_slot5 = ((1 - ptr5).*sigma + ptr5.*pds5.*Ts + ptr5.*(1 - pds5).*Tc);
s5 = pds5.*ptr5.*e./t_slot5;

% para n = 50
fun6 = @solucao_ah_v2_qk_n50;
result6 = fsolve(fun6,x0);
taud6 = result6(1)
b0_06 = result6(2)
n6 =50;

ptr6 = 1 - ((1-taud6)^n6_2slots);
pds6 = (n6_2slots*taud6*((1-taud6)^(n6_2slots-1)))/ptr6;
t_slot6 = ((1 - ptr6).*sigma + ptr6.*pds6.*Ts + ptr6.*(1 - pds6).*Tc);
s6 = pds6.*ptr6.*e./t_slot6;

% para n = 60
fun7 = @solucao_ah_v2_qk_n60;
result7 = fsolve(fun7,x0);
taud7 = result7(1)
b0_07 = result7(2)
n7 =60;

ptr7 = 1 - ((1-taud7)^n7_2slots);
pds7 = (n7_2slots*taud7*((1-taud7)^(n7_2slots-1)))/ptr7;
t_slot7 =((1 - ptr7).*sigma + ptr7.*pds7.*Ts + ptr7.*(1 - pds7).*Tc);
s7 = pds7.*ptr7.*e./t_slot7;

% para n = 70
fun8 = @solucao_ah_v2_qk_n70;
result8 = fsolve(fun8,x0);
taud8 =result8(1);
b0_08 = result8(2);
```

```
n8 =70;

ptr8 = 1 - ((1-taud8)^n8_2slots);
pds8 = (n8_2slots*taud8*((1-taud8)^(n8_2slots-1)))/ptr8;
t_slot8 = ((1 - ptr8).*sigma + ptr8.*pds8.*Ts + ptr8.*(1 - pds8).*Tc);
s8 = pds8.*ptr8.*e./t_slot8;

% para n = 80
fun9 = @solucao_ah_v2_qk_n80;
result9 = fsolve(fun9,x0);
taud9 = result9(1)
b0_09 = result9(2)
n9 = 80;

% cálculo das probabilidades
ptr9 = 1 - ((1-taud9)^n9_2slots);
pds9 = (n9_2slots*taud9*((1-taud9)^(n9_2slots-1)))/ptr9;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9 =((1 - ptr9).*sigma + ptr9.*pds9.*Ts + ptr9.*(1 - pds9).*Tc);
s9 = pds9.*ptr9.*e./t_slot9;

% 90 estações
fun10 = @solucao_ah_v2_qk_n90;
result10 = fsolve(fun10,x0);
taud10 = result10(1)
b0_010 = result10(2)
n10 =90;

ptr10 = 1 - ((1-taud10)^n10_2slots);
pds10 = (n10_2slots*taud10*((1-taud10)^(n10_2slots-1)))/ptr10;
t_slot10 = ((1 - ptr10).*sigma +
 ptr10.*pds10.*Ts + ptr10.*(1 - pds10).*Tc);
s10 = pds10.*ptr10.*e./t_slot10;

% para n=100
fun11 = @solucao_ah_v2_qk_n100;
result11 = fsolve(fun11,x0);
taud11 = result11(1)
b0_011 = result11(2)
n11 =100;

ptr11 = 1 - ((1-taud11)^n11_2slots);
pds11 = (n11_2slots*taud11*((1-taud11)^(n11_2slots-1)))/ptr11;
t_slot11 = ((1 - ptr11).*sigma +
 ptr11.*pds11.*Ts + ptr11.*(1 - pds11).*Tc)
s11 = (pds11.*ptr11.*e)./t_slot11;

vazao1_qk =(((Tidle_2)/beacon_interval)/1000000).*
[s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11].*2;

hold on
% plot(n,vazao1,'o-',n,vazao2,'*-.')
%plot(n,vazao1,'bo-',n,vazao1_razao,'bs-',n,vazao2,
'b*-',n,vazao2_idle,'bv-',nrawsta,vazao1ns3,'ro-')
%n,vazao2_idle,'bv-',
plot(n,vazao1_qkn,'bo-',n,vazao1_qk,'ks-',n,ns3_media2slots,'r*-')%,n,vazao1_razao_th,'bo-'
errorbar(n,ns3_media2slots,ns3_desvio2slots,'r')
% plot(n,vazao1_razao)
grid on
%title('q depende de tau e p - 2 slots')
%legend ('Nslot = 2','Nslot = 2 (razão)','Nslot = 2 (1-Pidle)',
'Nslot = 2 (Pidle*Tidle)','Nslot = 2 NS3')
legend ('Nslot = 2 modelo qkn','Nslot = 2 modelo qk',
'Nslot = 2 simulação NS3')%'Nslot = 2 (Th)',
xlabel('Número de estações')
ylabel('Vazão Mbit/s')


% comparacão entre o modelo analítico e a simulaçao no NS3 com a prob qk e qkn
% 5 slots RAW
% solução de tau-d e estados bi,j
clc
% solução de tau-d e estados bi,j
clc
% Definição de variáveis
basic_rate = 1000000; %bits/s
data_rate = 7800000; % bits/s
phy_header = 0.000192; %em segundos, parametro do ah
%h = 416/basic_rate; % (416 = 224 bits cabeçalho MAC + 192 bits cabeçalho
%PHY)em segundos, cabeçalho antigo
%ack = 304/basic_rate; % (304= 112 bits + 192 bits phy) em segundos
```

```
mac_header = (34*8)/basic_rate;
h = phy_header + mac_header;
ack = (14*8)/basic_rate+phy_header;
e = 256*8; %tamanho do payload/bitrate
e_duracao = 256*8/data_rate;
%sifs = 0.000160; % em segundos
%difs = 0.000303; % em segundos
%delta = 0.000006; % em segundos, estava 0.000001
%sigma = 0.000052; % em segundos
%ack_timeout = 0.000600;
sifs = 0.000160; % em segundos
difs = 0.000303; % em segundos
delta = 0.0000033; % em segundos, estava 0.000001
sigma = 0.000052; % em segundos
ack_timeout = 2*delta + sifs + ack;
beacon_interval = 0.1; %em segundos
%data_window_size = beacon_interval-20;
x0 = [0.01;0.01]; % valor inicial para fsolve
% tempo médio em que o canal está ocupado com uma transmissão com sucesso
Ts = h + e_duracao + sifs + (2.*delta) + ack + difs;
% tempo médio em que o canal está ocupado com uma trasmissão com colisão
Tc = h + e_duracao + difs + delta + ack_timeout;
Tg = 0.000008;
Th = Ts + 5*Tg;

%Tidle_2 = (beacon_interval/2);
Tidle_5 = (beacon_interval/5)- Th;
%Tidle_10 = (beacon_interval/10);
%Tidle_15 = (beacon_interval/15);

% para Nslot = 5 (5 slots dentro do raw)
% para n = 5
fun1_5slots = @solucao_ah_v5_qvariavel_n5;
result1_5slots = fsolve(fun1_5slots,x0);
taud1_5slots = result1_5slots(1)
b0_01_5slots = result1_5slots(2)
n1 = 5;
n1_5slots = 5/5;

% cálculo das probabilidades
ptr1_5slots = 1 - ((1-taud1_5slots)^n1_5slots);
pds1_5slots = (n1_5slots*taud1_5slots*((1-taud1_5slots)^(n1_5slots-1)))/ptr1_5slots;

% Cálculo das probabilidades de transmissão e de sucess
t_slot1_5slots = ((1 - ptr1_5slots).*sigma +
 ptr1_5slots.*pds1_5slots.*Ts + ptr1_5slots.*(1 - pds1_5slots).*Tc);
s1_5slots = pds1_5slots.*ptr1_5slots.*e./t_slot1_5slots;
b_idle5_1 = 0;
s1_5slots_idle = (pds1_5slots.*ptr1_5slots.*e./t_slot1_5slots)*(1 - b_idle5_1);
s1_5slots_idle1 = pds1_5slots.*ptr1_5slots.*e./(t_slot1_5slots + b_idle5_1*(1 - Tidle_5));

% para n = 10
fun2_5slots = @solucao_ah_v5_qvariavel_n10;
result2_5slots = fsolve(fun2_5slots,x0)
taud2_5slots = result2_5slots(1);
b0_02_5slots =result2_5slots(2);
n2 =10;
n2_5slots = n2/5;

ptr2_5slots = 1 - ((1-taud2_5slots)^n2_5slots);
pds2_5slots = (n2_5slots*taud2_5slots*((1-taud2_5slots)^(n2_5slots-1)))/ptr2_5slots;
t_slot2_5slots = ((1 - ptr2_5slots).*sigma +
ptr2_5slots.*pds2_5slots.*Ts + ptr2_5slots.*(1 - pds2_5slots).*Tc);
s2_5slots = pds2_5slots.*ptr2_5slots.*e./t_slot2_5slots;
b_idle5_2 = 0.0266;
s2_5slots_idle = (pds2_5slots.*ptr2_5slots.*e./t_slot2_5slots)*(1-b_idle5_2);
s2_5slots_idle1 = pds2_5slots.*ptr2_5slots.*e./(t_slot2_5slots + b_idle5_2*(1 - Tidle_5));

% para n = 20
fun3_5slots = @solucao_ah_v5_qvariavel_n20;
result3_5slots = fsolve(fun3_5slots,x0);
taud3_5slots = result3_5slots(1)
b0_03_5slots = result3_5slots(2)
n3 =20;
n3_5slots = n3/5;

ptr3_5slots = 1 - ((1-taud3_5slots)^n3_5slots);
pds3_5slots = (n3_5slots*taud3_5slots*((1-taud3_5slots)^(n3_5slots-1)))/ptr3_5slots;
t_slot3_5slots = ((1 - ptr3_5slots).*sigma +
 ptr3_5slots.*pds3_5slots.*Ts + ptr3_5slots.*(1 - pds3_5slots).*Tc);
s3_5slots = pds3_5slots.*ptr3_5slots.*e./t_slot3_5slots;
```

```
b_idle5_3 = 0.0299;
s3_5slots_idle = (pds3_5slots.*ptr3_5slots.*e./t_slot3_5slots)*(1 - b_idle5_3);
s3_5slots_idle1 = pds3_5slots.*ptr3_5slots.*e./(t_slot3_5slots + b_idle5_3*(1 - Tidle_5));

% para n = 30
fun4_5slots = @solucao_ah_v5_qvariavel_n30;
result4_5slots = fsolve(fun4_5slots,x0);
taud4_5slots = result4_5slots(1)
b0_04_5slots = result4_5slots(2)
n4 =30;
n4_5slots = n4/5;

ptr4_5slots = 1 - ((1-taud4_5slots)^n4_5slots);
pds4_5slots = (n4_5slots*taud4_5slots*((1-taud4_5slots)^(n4_5slots-1)))/ptr4_5slots;
t_slot4_5slots = ((1 - ptr4_5slots).*sigma +
 ptr4_5slots.*pds4_5slots.*Ts + ptr4_5slots.*(1 - pds4_5slots).*Tc);
s4_5slots = pds4_5slots.*ptr4_5slots.*e./t_slot4_5slots;
b_idle5_4 = 0.0313;
s4_5slots_idle = (pds4_5slots.*ptr4_5slots.*e./t_slot4_5slots)*(1-b_idle5_4);
s4_5slots_idle1 = pds4_5slots.*ptr4_5slots.*e./(t_slot4_5slots + b_idle5_4*(1 - Tidle_5));

% para n = 40
fun5_5slots = @solucao_ah_v5_qvariavel_n40;
result5_5slots = fsolve(fun5_5slots,x0);
taud5_5slots = result5_5slots(1)
b0_05_5slots = result5_5slots(2)
n5 =40;
n5_5slots = n5/5;

ptr5_5slots = 1 - ((1-taud5_5slots)^n5_5slots);
pds5_5slots = (n5_5slots*taud5_5slots*((1-taud5_5slots)^(n5_5slots-1)))/ptr5_5slots;
t_slot5_5slots = ((1 - ptr5_5slots).*sigma +
ptr5_5slots.*pds5_5slots.*Ts + ptr5_5slots.*(1 - pds5_5slots).*Tc);
s5_5slots = pds5_5slots.*ptr5_5slots.*e./t_slot5_5slots;
b_idle5_5 = 0.0322;
s5_5slots_idle = (pds5_5slots.*ptr5_5slots.*e./t_slot5_5slots)*(1 - b_idle5_5);
s5_5slots_idle1 = pds5_5slots.*ptr5_5slots.*e./(t_slot5_5slots + b_idle5_5*(1 - Tidle_5));
% para n = 50
fun6_5slots = @solucao_ah_v5_qvariavel_n50;
result6_5slots = fsolve(fun6_5slots,x0);
taud6_5slots = result6_5slots(1)
b0_06_5slots = result6_5slots(2)
n6 =50;
n6_5slots = n6/5;

ptr6_5slots = 1 - ((1-taud6_5slots)^n6_5slots);
pds6_5slots = (n6_5slots*taud6_5slots*((1-taud6_5slots)^(n6_5slots-1)))/ptr6_5slots;
t_slot6_5slots = ((1 - ptr6_5slots).*sigma
 + ptr6_5slots.*pds6_5slots.*Ts + ptr6_5slots.*(1 - pds6_5slots).*Tc);
s6_5slots = pds6_5slots.*ptr6_5slots.*e./t_slot6_5slots;
b_idle5_6 = 0.0328;
s6_5slots_idle = (pds6_5slots.*ptr6_5slots.*e./t_slot6_5slots)*(1 - b_idle5_6);
s6_5slots_idle1 = pds6_5slots.*ptr6_5slots.*e./(t_slot6_5slots + b_idle5_6*(1 - Tidle_5));

% para n = 60
fun7_5slots = @solucao_ah_v5_qvariavel_n60;
result7_5slots = fsolve(fun7_5slots,x0);
taud7_5slots = result7_5slots(1)
b0_07_5slots = result7_5slots(2)
n7 =60;
n7_5slots = n7/5;

ptr7_5slots = 1 - ((1-taud7_5slots)^n7_5slots);
pds7_5slots = (n7_5slots*taud7_5slots*((1-taud7_5slots)^(n7_5slots-1)))/ptr7_5slots;
t_slot7_5slots =((1 - ptr7_5slots).*sigma +
 ptr7_5slots.*pds7_5slots.*Ts + ptr7_5slots.*(1 - pds7_5slots).*Tc);
s7_5slots = pds7_5slots.*ptr7_5slots.*e./t_slot7_5slots;
b_idle5_7 = 0.0332;
s7_5slots_idle = (pds7_5slots.*ptr7_5slots.*e./t_slot7_5slots)*(1 - b_idle5_7);
s7_5slots_idle1 = pds7_5slots.*ptr7_5slots.*e./(t_slot7_5slots + b_idle5_7*(1 - Tidle_5));

% para n = 70
fun8_5slots = @solucao_ah_v5_qvariavel_n70;
result8_5slots = fsolve(fun8_5slots,x0);
taud8_5slots =result8_5slots(1)
b0_08_5slots = result8_5slots(2)
n8 =70;
n8_5slots = n8/5;

ptr8_5slots = 1 - ((1-taud8_5slots)^n8_5slots);
pds8_5slots = (n8_5slots*taud8_5slots*((1-taud8_5slots)^(n8_5slots-1)))/ptr8_5slots;
```

```
t_slot8_5slots = ((1 - ptr8_5slots).*sigma +
 ptr8_5slots.*pds8_5slots.*Ts + ptr8_5slots.*(1 - pds8_5slots).*Tc);
s8_5slots = pds8_5slots.*ptr8_5slots.*e./t_slot8_5slots;
b_idle5_8 = 0.0344;
s8_5slots_idle = (pds8_5slots.*ptr8_5slots.*e./t_slot8_5slots)*(1 - b_idle5_8);
s8_5slots_idle1 = pds8_5slots.*ptr8_5slots.*e./(t_slot8_5slots + b_idle5_8*(1 - Tidle_5));

% para n = 80
fun9_5slots = @solucao_ah_v5_qvariavel_n80;
result9_5slots = fsolve(fun9_5slots,x0);
taud9_5slots = result9_5slots(1)
b0_09_5slots = result9_5slots(2)
n9 = 80;
n9_5slots = n9/5;

% cálculo das probabilidades
ptr9_5slots = 1 - ((1-taud9_5slots)^n9_5slots);
pds9_5slots = (n9_5slots*taud9_5slots*((1-taud9_5slots)^(n9_5slots-1)))/ptr9_5slots;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9_5slots =((1 - ptr9_5slots).*sigma +
 ptr9_5slots.*pds9_5slots.*Ts + ptr9_5slots.*(1 - pds9_5slots).*Tc);
s9_5slots = pds9_5slots.*ptr9_5slots.*e./t_slot9_5slots;
b_idle5_9 = 0.0336;
s9_5slots_idle = (pds9_5slots.*ptr9_5slots.*e./t_slot9_5slots)*(1 - b_idle5_9);
s9_5slots_idle1 = pds9_5slots.*ptr9_5slots.*e./(t_slot9_5slots + b_idle5_9*(1 - Tidle_5));

% 90 estações
fun10_5slots = @solucao_ah_v5_qvariavel_n90;
result10_5slots = fsolve(fun10_5slots,x0);
taud10_5slots = result10_5slots(1)
b0_010_5slots = result10_5slots(2)
n10 =90;
n10_5slots = n10/5;

ptr10_5slots = 1 - ((1-taud10_5slots)^n10_5slots);
pds10_5slots = (n10_5slots*taud10_5slots*((1-taud10_5slots)^(n10_5slots-1)))/ptr10_5slots;
t_slot10_5slots = ((1 - ptr10_5slots).*sigma +
 ptr10_5slots.*pds10_5slots.*Ts + ptr10_5slots.*(1 - pds10_5slots).*Tc);
s10_5slots = pds10_5slots.*ptr10_5slots.*e./t_slot10_5slots;
b_idle5_10 = 0.0337;
s10_5slots_idle = (pds10_5slots.*ptr10_5slots.*e./t_slot10_5slots)*(1-b_idle5_10);
s10_5slots_idle1 = pds10_5slots.*ptr10_5slots.*e./(t_slot10_5slots + b_idle5_10*(1 - Tidle_5));

% para n=100
fun11_5slots = @solucao_ah_v5_qvariavel_n100;
result11_5slots = fsolve(fun11_5slots,x0);
taud11_5slots = result11_5slots(1)
b0_011_5slots = result11_5slots(2)
n11 =100;
n11_5slots = 100/5;

ptr11_5slots = 1 - ((1-taud11_5slots)^n11_5slots);
pds11_5slots = (n11_5slots*taud11_5slots*((1-taud11_5slots)^(n11_5slots-1)))/ptr11_5slots;
t_slot11_5slots = ((1 - ptr11_5slots).*sigma +
 ptr11_5slots.*pds11_5slots.*Ts + ptr11_5slots.*(1 - pds11_5slots).*Tc)
s11_5slots = (pds11_5slots.*ptr11_5slots.*e)./t_slot11_5slots;
b_idle5_11 = 0.0338;
s11_5slots_idle = ((pds11_5slots.*ptr11_5slots.*e)./t_slot11_5slots)*(1 - b_idle5_11);
s11_5slots_idle1 = pds11_5slots.*ptr11_5slots.*e./(t_slot11_5slots + b_idle5_11*(1 - Tidle_5));


n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11];
vazao2 =(((Tidle_5)/beacon_interval)/1000000).*[s2_5slots,s2_5slots,s3_5slots,
s4_5slots,s5_5slots,s6_5slots,s7_5slots,s8_5slots,s9_5slots,s10_5slots,s11_5slots].*5;
vazao2_qkn =((Tidle_5/beacon_interval)/1000000).*[s2_5slots-0.0002,s2_5slots,s3_5slots,
s4_5slots,s5_5slots,s6_5slots,s7_5slots,s8_5slots,s9_5slots,s10_5slots,s11_5slots].*5;

%vazão qk
% para Nslot = 5 (10 slots dentro do raw)
% para n = 5
fun1 = @solucao_ah_v5_qk_n5;
result1 = fsolve(fun1,x0);
taud1 = result1(1);
b0_01 = result1(2);
n1 = 5;

% cálculo das probabilidades
ptr1 = 1 - ((1-taud1)^n1_5slots);
pds1 = (n1_5slots*taud1*((1-taud1)^(n1_5slots-1)))/ptr1;

% Cálculo das probabilidades de transmissão e de sucess
```

```
t_slot1 = ((1 - ptr1).*sigma + ptr1.*pds1.*Ts + ptr1.*(1 - pds1).*Tc);
s1 = (pds1.*ptr1.*e./t_slot1);

% para n = 10
fun2 = @solucao_ah_v5_qk_n10;
result2 = fsolve(fun2,x0)
taud2 = result2(1);
b0_02 =result2(2);

ptr2 = 1 - ((1-taud2)^n2_5slots);
pds2 = (n2_5slots*taud2*((1-taud2)^(n2_5slots-1)))/ptr2;
t_slot2 = ((1 - ptr2).*sigma + ptr2.*pds2.*Ts + ptr2.*(1 - pds2).*Tc);
s2 = (pds2.*ptr2.*e./t_slot2);

% para n = 20
fun3 = @solucao_ah_v5_qk_n20;
result3 = fsolve(fun3,x0);
taud3 = result3(1)
b0_03 = result3(2)
n3 =20;

ptr3 = 1 - ((1-taud3)^n3_5slots);
pds3 = (n3_5slots*taud3*((1-taud3)^(n3_5slots-1)))/ptr3;
t_slot3 = ((1 - ptr3).*sigma + ptr3.*pds3.*Ts + ptr3.*(1 - pds3).*Tc);
s3 = pds3.*ptr3.*e./t_slot3;

% para n = 30
fun4 = @solucao_ah_v5_qk_n30;
result4 = fsolve(fun4,x0);
taud4 = result4(1);
b0_04 = result4(2);
n4 =30;

ptr4 = 1 - ((1-taud4)^n4_5slots);
pds4 = (n4_5slots*taud4*((1-taud4)^(n4_5slots-1)))/ptr4;
t_slot4 = ((1 - ptr4).*sigma + ptr4.*pds4.*Ts + ptr4.*(1 - pds4).*Tc);
s4 = pds4.*ptr4.*e./t_slot4;

% para n = 40
fun5 = @solucao_ah_v5_qk_n40;
result5 = fsolve(fun5,x0);
taud5 = result5(1)
b0_05 = result5(2)
n5 =40;

ptr5 = 1 - ((1-taud5)^n5_5slots);
pds5 = (n5_5slots*taud5*((1-taud5)^(n5_5slots-1)))/ptr5;
t_slot5 = ((1 - ptr5).*sigma + ptr5.*pds5.*Ts + ptr5.*(1 - pds5).*Tc);
s5 = pds5.*ptr5.*e./t_slot5;

% para n = 50
fun6 = @solucao_ah_v5_qk_n50;
result6 = fsolve(fun6,x0);
taud6 = result6(1)
b0_06 = result6(2)
n6 =50;

ptr6 = 1 - ((1-taud6)^n6_5slots);
pds6 = (n6_5slots*taud6*((1-taud6)^(n6_5slots-1)))/ptr6;
t_slot6 = ((1 - ptr6).*sigma + ptr6.*pds6.*Ts + ptr6.*(1 - pds6).*Tc);
s6 = pds6.*ptr6.*e./t_slot6;

% para n = 60
fun7 = @solucao_ah_v5_qk_n60;
result7 = fsolve(fun7,x0);
taud7 = result7(1)
b0_07 = result7(2)
n7 =60;

ptr7 = 1 - ((1-taud7)^n7_5slots);
pds7 = (n7_5slots*taud7*((1-taud7)^(n7_5slots-1)))/ptr7;
t_slot7 =((1 - ptr7).*sigma + ptr7.*pds7.*Ts + ptr7.*(1 - pds7).*Tc);
s7 = pds7.*ptr7.*e./t_slot7;

% para n = 70
fun8 = @solucao_ah_v5_qk_n70;
result8 = fsolve(fun8,x0);
taud8 =result8(1);
b0_08 = result8(2);
n8 =70;
```

```
ptr8 = 1 - ((1-taud8)^n8_5slots);
pds8 = (n8_5slots*taud8*((1-taud8)^(n8_5slots-1)))/ptr8;
t_slot8 = ((1 - ptr8).*sigma + ptr8.*pds8.*Ts + ptr8.*(1 - pds8).*Tc);
s8 = pds8.*ptr8.*e./t_slot8;

% para n = 80
fun9 = @solucao_ah_v5_qk_n80;
result9 = fsolve(fun9,x0);
taud9 = result9(1)
b0_09 = result9(2)
n9 = 80;

% cálculo das probabilidades
ptr9 = 1 - ((1-taud9)^n9_5slots);
pds9 = (n9_5slots*taud9*((1-taud9)^(n9_5slots-1)))/ptr9;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9 =((1 - ptr9).*sigma + ptr9.*pds9.*Ts + ptr9.*(1 - pds9).*Tc);
s9 = pds9.*ptr9.*e./t_slot9;

% 90 estações
fun10 = @solucao_ah_v5_qk_n90;
result10 = fsolve(fun10,x0);
taud10 = result10(1)
b0_010 = result10(2)
n10 =90;

ptr10 = 1 - ((1-taud10)^n10_5slots);
pds10 = (n10_5slots*taud10*((1-taud10)^(n10_5slots-1)))/ptr10;
t_slot10 = ((1 - ptr10).*sigma + ptr10.*pds10.*Ts + ptr10.*(1 - pds10).*Tc);
s10 = pds10.*ptr10.*e./t_slot10;

% para n=100
fun11 = @solucao_ah_v5_qk_n100;
result11 = fsolve(fun11,x0);
taud11 = result11(1)
b0_011 = result11(2)
n11 =100;

ptr11 = 1 - ((1-taud11)^n11_5slots);
pds11 = (n11_5slots*taud11*((1-taud11)^(n11_5slots-1)))/ptr11;
t_slot11 = ((1 - ptr11).*sigma + ptr11.*pds11.*Ts + ptr11.*(1 - pds11).*Tc)
s11 = (pds11.*ptr11.*e)./t_slot11;


%vazao2_qk =(((Tidle_5)/beacon_interval)/1000000).*[s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11].*5;

simulacaons3_5slots;

%(Tidle_5/beacon_interval)
% 5 slots
nrawsta=[5,10,20,30,40,50,60,70,80,90,100];

figure(1)
hold on
% plot(n,vazao1,'o-',n,vazao2,'*-.')
% plot(n,vazao2,'bo-',n,vazao2_razao,'bs-',n,vazao2_idle,'b*-',n,vazao2_idle1,'bv-',nrawsta,vazao2ns3,'ro-')
plot(n,vazao2_qkn,'bo-',n,vazao2_qk,'ks-',n,ns3_media5slots,'r*-')%,n,vazao1_razao_th,'bo-'
errorbar(n,ns3_media5slots,ns3_desvio5slots,'r')%n,vazao2_razao,'bo-',
%plot(n,vazao4)
grid on
%title('q dependendo de tau e p - 5 slots')
% legend ('Nslot = 5','Nslot = 5 (razão)','Nslot = 5 (1-Pidle)','Nslot = 5 (Pidle*Tidle)','Nslot = 5 NS3')
legend ('Nslot = 5 modelo qkn','Nslot = 5 modelo qk','Nslot = 5 simulação NS3')%'Nslot = 5 (Th)',
xlabel('Número de estações')
ylabel('Vazão Mbit/s')


 % comparação modelo e ns3
 %10 slots RAW

clc
% Definição de variáveis
basic_rate = 1000000; %bits/s
data_rate = 7800000; % bits/s
phy_header = 0.000192; %em segundos, parametro do ah
%h = 416/basic_rate; % (416 = 224 bits cabeçalho MAC + 192 bits cabeçalho
%PHY)em segundos, cabeçalho antigo
%ack = 304/basic_rate; % (304= 112 bits + 192 bits phy) em segundos
mac_header = (34*8)/basic_rate;
h = phy_header + mac_header;
ack = (14*8)/basic_rate + phy_header;
```

```
e = 256*8; %tamanho do payload/bitrate
e_duracao = 256*8/data_rate;
%sifs = 0.000160; % em segundos
%difs = 0.000303; % em segundos
%delta = 0.000006; % em segundos, estava 0.000001
%sigma = 0.000052; % em segundos
%ack_timeout = 0.000600;
sifs = 0.000160; % em segundos
difs = 0.000120; % em segundos
delta = 0.0000033; % em segundos, estava 0.000001
sigma = 0.000052; % em segundos
ack_timeout = 2*delta + sifs + ack;
beacon_interval = 0.1; %em segundos

%data_window_size = beacon_interval-20;
x0 = [0.01;0.01]; % valor inicial para fsolve
% tempo médio em que o canal está ocupado com uma transmissão com sucesso
Ts = h + e_duracao + sifs + (2.*delta) + ack + difs;
% tempo médio em que o canal está ocupado com uma trasmissão com colisão
Tc = h + e_duracao + difs + delta + ack_timeout;
Tg = 0.000008;
Th = Ts + 10*Tg ;

%Tidle_2 = (beacon_interval/2);
%Tidle_5 = (beacon_interval/5);
Tidle_10 = (beacon_interval/10)-Th;
%Tidle_15 = (beacon_interval/15);

% para Nslot = 10 (10 slots dentro do raw)
% para n = 5
fun1_10slots = @solucao_ah_v10_qvariavel_n5;
result1_10slots = fsolve(fun1_10slots,x0);
taud1_10slots = result1_10slots(1)
b0_01_10slots = result1_10slots(2)
n1 = 5;
n1_10slots = 1;

% cálculo das probabilidades
ptr1_10slots = 1 - ((1-taud1_10slots)^n1_10slots);
pds1_10slots = (n1_10slots*taud1_10slots*((1-taud1_10slots)^(n1_10slots-1)))/ptr1_10slots;

% Cálculo das probabilidades de transmissão e de sucess
t_slot1_10slots = ((1 - ptr1_10slots).*sigma +
ptr1_10slots.*pds1_10slots.*Ts + ptr1_10slots.*(1 - pds1_10slots).*Tc);
s1_10slots = (pds1_10slots.*ptr1_10slots.*e./t_slot1_10slots) - 0.001 ;

% para n = 10
fun2_10slots = @solucao_ah_v10_qvariavel_n10;
result2_10slots = fsolve(fun2_10slots,x0)
taud2_10slots = result2_10slots(1);
b0_02_10slots =result2_10slots(2);
n2 =10;
n2_10slots = n2/10;

ptr2_10slots = 1 - ((1-taud2_10slots)^n2_10slots);
pds2_10slots = (n2_10slots*taud2_10slots*((1-taud2_10slots)^(n2_10slots-1)))/ptr2_10slots;
t_slot2_10slots = ((1 - ptr2_10slots).*sigma +
ptr2_10slots.*pds2_10slots.*Ts + ptr2_10slots.*(1 - pds2_10slots).*Tc);
s2_10slots = (pds2_10slots.*ptr2_10slots.*e./t_slot2_10slots);

% para n = 20
fun3_10slots = @solucao_ah_v10_qvariavel_n20;
result3_10slots = fsolve(fun3_10slots,x0);
taud3_10slots = result3_10slots(1)
b0_03_10slots = result3_10slots(2)
n3 =20;
n3_10slots = n3/10;

ptr3_10slots = 1 - ((1-taud3_10slots)^n3_10slots);
pds3_10slots = (n3_10slots*taud3_10slots*((1-taud3_10slots)^(n3_10slots-1)))/ptr3_10slots;
t_slot3_10slots = ((1 - ptr3_10slots).*sigma +
 ptr3_10slots.*pds3_10slots.*Ts + ptr3_10slots.*(1 - pds3_10slots).*Tc);
s3_10slots = pds3_10slots.*ptr3_10slots.*e./t_slot3_10slots;

% para n = 30
fun4_10slots = @solucao_ah_v10_qvariavel_n30;
result4_10slots = fsolve(fun4_10slots,x0);
taud4_10slots = result4_10slots(1)
b0_04_10slots = result4_10slots(2)
n4 =30;
n4_10slots = n4/10;
```

```
ptr4_10slots = 1 - ((1-taud4_10slots)^n4_10slots);
pds4_10slots = (n4_10slots*taud4_10slots*((1-taud4_10slots)^(n4_10slots-1)))/ptr4_10slots;
t_slot4_10slots = ((1 - ptr4_10slots).*sigma +
ptr4_10slots.*pds4_10slots.*Ts + ptr4_10slots.*(1 - pds4_10slots).*Tc);
s4_10slots = pds4_10slots.*ptr4_10slots.*e./t_slot4_10slots;

% para n = 40
fun5_10slots = @solucao_ah_v10_qvariavel_n40;
result5_10slots = fsolve(fun5_10slots,x0);
taud5_10slots = result5_10slots(1)
b0_05_10slots = result5_10slots(2)
n5 =40;
n5_10slots = n5/10;

ptr5_10slots = 1 - ((1-taud5_10slots)^n5_10slots);
pds5_10slots = (n5_10slots*taud5_10slots*((1-taud5_10slots)^(n5_10slots-1)))/ptr5_10slots;
t_slot5_10slots = ((1 - ptr5_10slots).*sigma +
 ptr5_10slots.*pds5_10slots.*Ts + ptr5_10slots.*(1 - pds5_10slots).*Tc);
s5_10slots = pds5_10slots.*ptr5_10slots.*e./t_slot5_10slots;

% para n = 50
fun6_10slots = @solucao_ah_v10_qvariavel_n50;
result6_10slots = fsolve(fun6_10slots,x0);
taud6_10slots = result6_10slots(1)
b0_06_10slots = result6_10slots(2)
n6 =50;
n6_10slots = n6/10;

ptr6_10slots = 1 - ((1-taud6_10slots)^n6_10slots);
pds6_10slots = (n6_10slots*taud6_10slots*((1-taud6_10slots)^(n6_10slots-1)))/ptr6_10slots;
t_slot6_10slots = ((1 - ptr6_10slots).*sigma +
ptr6_10slots.*pds6_10slots.*Ts + ptr6_10slots.*(1 - pds6_10slots).*Tc);
s6_10slots = pds6_10slots.*ptr6_10slots.*e./t_slot6_10slots;

% para n = 60
fun7_10slots = @solucao_ah_v10_qvariavel_n60;
result7_10slots = fsolve(fun7_10slots,x0);
taud7_10slots = result7_10slots(1)
b0_07_10slots = result7_10slots(2)
n7 =60;
n7_10slots = n7/10;

ptr7_10slots = 1 - ((1-taud7_10slots)^n7_10slots);
pds7_10slots = (n7_10slots*taud7_10slots*((1-taud7_10slots)^(n7_10slots-1)))/ptr7_10slots;
t_slot7_10slots =((1 - ptr7_10slots).*sigma +
 ptr7_10slots.*pds7_10slots.*Ts + ptr7_10slots.*(1 - pds7_10slots).*Tc);
s7_10slots = pds7_10slots.*ptr7_10slots.*e./t_slot7_10slots;

% para n = 70
fun8_10slots = @solucao_ah_v10_qvariavel_n70;
result8_10slots = fsolve(fun8_10slots,x0);
taud8_10slots =result8_10slots(1)
b0_08_10slots = result8_10slots(2)
n8 =70;
n8_10slots = n8/10;

ptr8_10slots = 1 - ((1-taud8_10slots)^n8_10slots);
pds8_10slots = (n8_10slots*taud8_10slots*((1-taud8_10slots)^(n8_10slots-1)))/ptr8_10slots;
t_slot8_10slots = ((1 - ptr8_10slots).*sigma +
ptr8_10slots.*pds8_10slots.*Ts + ptr8_10slots.*(1 - pds8_10slots).*Tc);
s8_10slots = pds8_10slots.*ptr8_10slots.*e./t_slot8_10slots;

% para n = 80
fun9_10slots = @solucao_ah_v10_qvariavel_n80;
result9_10slots = fsolve(fun9_10slots,x0);
taud9_10slots = result9_10slots(1)
b0_09_10slots = result9_10slots(2)
n9 = 80;
n9_10slots = n9/10;

% cálculo das probabilidades
ptr9_10slots = 1 - ((1-taud9_10slots)^n9_10slots);
pds9_10slots = (n9_10slots*taud9_10slots*((1-taud9_10slots)^(n9_10slots-1)))/ptr9_10slots;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9_10slots =((1 - ptr9_10slots).*sigma +
 ptr9_10slots.*pds9_10slots.*Ts + ptr9_10slots.*(1 - pds9_10slots).*Tc);
s9_10slots = pds9_10slots.*ptr9_10slots.*e./t_slot9_10slots;

% 90 estações
fun10_10slots = @solucao_ah_v10_qvariavel_n90;
```

```
result10_10slots = fsolve(fun10_10slots,x0);
taud10_10slots = result10_10slots(1)
b0_010_10slots = result10_10slots(2)
n10 =90;
n10_10slots = n10/10;

ptr10_10slots = 1 - ((1-taud10_10slots)^n10_10slots);
pds10_10slots = (n10_10slots*taud10_10slots*((1-taud10_10slots)^(n10_10slots-1)))/ptr10_10slots;
t_slot10_10slots = ((1 - ptr10_10slots).*sigma +
 ptr10_10slots.*pds10_10slots.*Ts + ptr10_10slots.*(1 - pds10_10slots).*Tc);
s10_10slots = pds10_10slots.*ptr10_10slots.*e./t_slot10_10slots;

% para n=100
fun11_10slots = @solucao_ah_v10_qvariavel_n100;
result11_10slots = fsolve(fun11_10slots,x0);
taud11_10slots = result11_10slots(1)
b0_011_10slots = result11_10slots(2)
n11 =100;
n11_10slots = 100/10;

ptr11_10slots = 1 - ((1-taud11_10slots)^n11_10slots);
pds11_10slots = (n11_10slots*taud11_10slots*((1-taud11_10slots)^(n11_10slots-1)))/ptr11_10slots;
t_slot11_10slots = ((1 - ptr11_10slots).*sigma +
 ptr11_10slots.*pds11_10slots.*Ts + ptr11_10slots.*(1 - pds11_10slots).*Tc)
s11_10slots = (pds11_10slots.*ptr11_10slots.*e)./t_slot11_10slots;

%vazão qk
% para Nslot = 10 (10 slots dentro do raw)
% para n = 5
fun1 = @solucao_ah_v10_qk_n5;
result1 = fsolve(fun1,x0);
taud1 = result1(1);
b0_01 = result1(2);
n1 = 5;

% cálculo das probabilidades
ptr1 = 1 - ((1-taud1)^n1_10slots);
pds1 = (n1_10slots*taud1*((1-taud1)^(n1_10slots-1)))/ptr1;

% Cálculo das probabilidades de transmissão e de sucess
t_slot1 = ((1 - ptr1).*sigma + ptr1.*pds1.*Ts + ptr1.*(1 - pds1).*Tc);
s1 = (pds1.*ptr1.*e./t_slot1);

% para n = 10
fun2 = @solucao_ah_v10_qk_n10;
result2 = fsolve(fun2,x0)
taud2 = result2(1);
b0_02 =result2(2);

ptr2 = 1 - ((1-taud2)^n2_10slots);
pds2 = (n2_10slots*taud2*((1-taud2)^(n2_10slots-1)))/ptr2;
t_slot2 = ((1 - ptr2).*sigma + ptr2.*pds2.*Ts + ptr2.*(1 - pds2).*Tc);
s2 = (pds2.*ptr2.*e./t_slot2);

% para n = 20
fun3 = @solucao_ah_v10_qk_n20;
result3 = fsolve(fun3,x0);
taud3 = result3(1)
b0_03 = result3(2)
n3 =20;

ptr3 = 1 - ((1-taud3)^n3_10slots);
pds3 = (n3_10slots*taud3*((1-taud3)^(n3_10slots-1)))/ptr3;
t_slot3 = ((1 - ptr3).*sigma + ptr3.*pds3.*Ts + ptr3.*(1 - pds3).*Tc);
s3 = pds3.*ptr3.*e./t_slot3;

% para n = 30
fun4 = @solucao_ah_v10_qk_n30;
result4 = fsolve(fun4,x0);
taud4 = result4(1);
b0_04 = result4(2);
n4 =30;

ptr4 = 1 - ((1-taud4)^n4_10slots);
pds4 = (n4_10slots*taud4*((1-taud4)^(n4_10slots-1)))/ptr4;
t_slot4 = ((1 - ptr4).*sigma + ptr4.*pds4.*Ts + ptr4.*(1 - pds4).*Tc);
s4 = pds4.*ptr4.*e./t_slot4;

% para n = 40
fun5 = @solucao_ah_v10_qk_n40;
result5 = fsolve(fun5,x0);
```

```
taud5 = result5(1)
b0_05 = result5(2)
n5 =40;

ptr5 = 1 - ((1-taud5)^n5_10slots);
pds5 = (n5_10slots*taud5*((1-taud5)^(n5_10slots-1)))/ptr5;
t_slot5 = ((1 - ptr5).*sigma + ptr5.*pds5.*Ts + ptr5.*(1 - pds5).*Tc);
s5 = pds5.*ptr5.*e./t_slot5;

% para n = 50
fun6 = @solucao_ah_v10_qk_n50;
result6 = fsolve(fun6,x0);
taud6 = result6(1)
b0_06 = result6(2)
n6 =50;

ptr6 = 1 - ((1-taud6)^n6_10slots);
pds6 = (n6_10slots*taud6*((1-taud6)^(n6_10slots-1)))/ptr6;
t_slot6 = ((1 - ptr6).*sigma + ptr6.*pds6.*Ts + ptr6.*(1 - pds6).*Tc);
s6 = pds6.*ptr6.*e./t_slot6;

% para n = 60
fun7 = @solucao_ah_v10_qk_n60;
result7 = fsolve(fun7,x0);
taud7 = result7(1)
b0_07 = result7(2)
n7 =60;

ptr7 = 1 - ((1-taud7)^n7_10slots);
pds7 = (n7_10slots*taud7*((1-taud7)^(n7_10slots-1)))/ptr7;
t_slot7 =((1 - ptr7).*sigma + ptr7.*pds7.*Ts + ptr7.*(1 - pds7).*Tc);
s7 = pds7.*ptr7.*e./t_slot7;

% para n = 70
fun8 = @solucao_ah_v10_qk_n70;
result8 = fsolve(fun8,x0);
taud8 =result8(1);
b0_08 = result8(2);
n8 =70;

ptr8 = 1 - ((1-taud8)^n8_10slots);
pds8 = (n8_10slots*taud8*((1-taud8)^(n8_10slots-1)))/ptr8;
t_slot8 = ((1 - ptr8).*sigma + ptr8.*pds8.*Ts + ptr8.*(1 - pds8).*Tc);
s8 = pds8.*ptr8.*e./t_slot8;

% para n = 80
fun9 = @solucao_ah_v10_qk_n80;
result9 = fsolve(fun9,x0);
taud9 = result9(1)
b0_09 = result9(2)
n9 = 80;

% cálculo das probabilidades
ptr9 = 1 - ((1-taud9)^n9_10slots);
pds9 = (n9_10slots*taud9*((1-taud9)^(n9_10slots-1)))/ptr9;
% Cálculo das probabilidades de transmissão e de sucess
t_slot9 =((1 - ptr9).*sigma +
 ptr9.*pds9.*Ts + ptr9.*(1 - pds9).*Tc);
s9 = pds9.*ptr9.*e./t_slot9;

% 90 estações
fun10 = @solucao_ah_v10_qk_n90;
result10 = fsolve(fun10,x0);
taud10 = result10(1)
b0_010 = result10(2)
n10 =90;

ptr10 = 1 - ((1-taud10)^n10_10slots);
pds10 = (n10_10slots*taud10*((1-taud10)^(n10_10slots-1)))/ptr10;
t_slot10 = ((1 - ptr10).*sigma +
 ptr10.*pds10.*Ts + ptr10.*(1 - pds10).*Tc);
s10 = pds10.*ptr10.*e./t_slot10;

% para n=100
fun11 = @solucao_ah_v10_qk_n100;
result11 = fsolve(fun11,x0);
taud11 = result11(1)
b0_011 = result11(2)
n11 =100;

ptr11 = 1 - ((1-taud11)^n11_10slots);
```

```
pds11 = (n11_10slots*taud11*((1-taud11)^(n11_10slots-1)))/ptr11;
t_slot11 = ((1 - ptr11).*sigma + ptr11.*pds11.*Ts + ptr11.*(1 - pds11).*Tc)
s11 = (pds11.*ptr11.*e)./t_slot11;

vazao3_qk =(((Tidle_10)/beacon_interval)/1000000).*[5*s1,10*s2,10*s3,
10*s4,10*s5,10*s6,10*s7,10*s8,10*s9,10*s10,10*s11];


n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11];
vazao3_qkn =(((Tidle_10)/beacon_interval)/1000000).*[5*s1_10slots,10*s2_10slots,10*s3_10slots,
10*s4_10slots,10*s5_10slots,10*s6_10slots,10*s7_10slots,10*s8_10slots,
10*s9_10slots,10*s10_10slots,10*s11_10slots];


simulacaons3_10slots;

nrawsta=[5,10,20,30,40,50,60,70,80,90,100];


figure(1)
hold on
% plot(n,vazao1,'o-',n,vazao2,'*-.')
%plot(n,vazao2,'bo-',n,vazao2_razao,'bs-',n,vazao2_idle,'b*-',n,vazao2_idle1,'bv-',nrawsta,vazao2ns3,'ro-')
plot(n,vazao3_qkn,'bo-',n,vazao3_qk,'ks-',nrawsta,ns3_media10slots,'r*-')%,n,vazao1_razao_th,'bo-'
errorbar(nrawsta,ns3_media10slots,ns3_desvio10slots,'r')
%plot(n,vazao4)
grid on
% title('q dependendo de tau e p - 10 slots')
%legend ('Nslot = 10','Nslot = 10 (razão)','Nslot = 10 (1-Pidle)','Nslot = 10 (Pidle*Tidle)','Nslot = 10 NS3')
legend ('Nslot = 10 modelo qkn','Nslot = 10 modelo qk','Nslot = 10 simulação NS3')%,'Nslot = 10 (Th)'
xlabel('Número de estações')
ylabel('Vazão Mbit/s')


% comparacão entre o modelo analítico e a simulaçao no NS3 com a prob q
% constante
% vazão calculada conforme o simulador faria
% solução de tau-d e estados bi,j
clc
% Definição de variáveis
basic_rate = 1000000; %bits/s
data_rate = 7800000; % bits/s
phy_header = 0.000192; %em segundos, parametro do ah
%h = 416/basic_rate; % (416 = 224 bits cabeçalho MAC + 192 bits cabeçalho
%PHY)em segundos, cabeçalho antigo
%ack = 304/basic_rate; % (304= 112 bits + 192 bits phy) em segundos
mac_header = (34*8)/basic_rate;
h = phy_header + mac_header;
ack = (14*8)/basic_rate + phy_header;
e = 256*8; %tamanho do payload/bitrate
e_duracao = 256*8/data_rate;
sifs = 0.000160; % em segundos
difs = 0.001120; % em segundos
delta = 0.0000033; % em segundos, estava 0.000001
sigma = 0.000052; % em segundos
ack_timeout = 2*delta + sifs +ack;
beacon_interval = 0.1; %em segundos
%data_window_size = beacon_interval-20;
x0 = [0.01;0.01]; % valor inicial para fsolve
% tempo médio em que o canal está ocupado com uma transmissão com sucesso
Ts = h + e_duracao + sifs + (2.*delta) + ack;
% tempo médio em que o canal está ocupado com uma trasmissão com colisão
Tc = h + e_duracao + difs + delta + ack_timeout;% NO LUGAR DE DIFS COLOCAR EIFS
Tg = 0.000008;
Th = Ts + 14*Tg;
Tidle_15 = (beacon_interval/15) - Th;

n1 = 5;
n2 = 10;
n3 = 20;
n4 = 30;
n5 = 40;
n6 = 50;
n7 = 60;
n8 = 70;
n9 = 80;
n10 = 90;
n11 = 100;

% para Nslot = 15 (15 slots dentro do raw)
% para n = 1
```

```
fun1_15slots = @solucao_ah_v15_qvariavel_n1;
result1_15slots = fsolve(fun1_15slots,x0);
taud1_15slots = 1;%result1_15slots(1)
b0_01_15slots =1;%result1_15slots(2)
n1_15slots = 1;

% cálculo das probabilidades
ptr1_15slots = 1 - ((1-taud1_15slots)^n1_15slots);
pds1_15slots = (n1_15slots*taud1_15slots*((1-taud1_15slots)^(n1_15slots-1)))/ptr1_15slots;

% Cálculo das probabilidades de transmissão e de sucess
t_slot1_15slots =ptr1_15slots.*pds1_15slots.*Ts %((1 - ptr1_15slots).*sigma
+ ptr1_15slots.*pds1_15slots.*Ts + ptr1_15slots.*(1 - pds1_15slots).*Tc);
s1_15slots =  pds1_15slots.*ptr1_15slots.*e./t_slot1_15slots

% para n = 2
fun2_15slots = @solucao_ah_v15_qvariavel_n2;
result2_15slots = fsolve(fun2_15slots,x0)
taud2_15slots = result2_15slots(1);
b0_02_15slots =result2_15slots(2);
n2_15slots = 2;

ptr2_15slots = 1 - ((1-taud2_15slots)^n2_15slots);
pds2_15slots = (n2_15slots*taud2_15slots*((1-taud2_15slots)^(n2_15slots-1)))/ptr2_15slots;
t_slot2_15slots = ((1 - ptr2_15slots).*sigma +
 ptr2_15slots.*pds2_15slots.*Ts + ptr2_15slots.*(1 - pds2_15slots).*Tc);
s2_15slots = pds2_15slots.*ptr2_15slots.*e./t_slot2_15slots

% para n = 3
fun3_15slots = @solucao_ah_v15_qvariavel_n3;
result3_15slots = fsolve(fun3_15slots,x0);
taud3_15slots = result3_15slots(1)
b0_03_15slots = result3_15slots(2)
n3_15slots = 3;

ptr3_15slots = 1 - ((1-taud3_15slots)^n3_15slots);
pds3_15slots = (n3_15slots*taud3_15slots*((1-taud3_15slots)^(n3_15slots-1)))/ptr3_15slots;
t_slot3_15slots = ((1 - ptr3_15slots).*sigma +
 ptr3_15slots.*pds3_15slots.*Ts + ptr3_15slots.*(1 - pds3_15slots).*Tc);
s3_15slots = pds3_15slots.*ptr3_15slots.*e./t_slot3_15slots

% para n = 4
fun4_15slots = @solucao_ah_v15_qvariavel_n4;
result4_15slots = fsolve(fun4_15slots,x0);
taud4_15slots = result4_15slots(1)
b0_04_15slots = result4_15slots(2)
n4_15slots = 4;

ptr4_15slots = 1 - ((1-taud4_15slots)^n4_15slots);
pds4_15slots = (n4_15slots*taud4_15slots*((1-taud4_15slots)^(n4_15slots-1)))/ptr4_15slots;
t_slot4_15slots = ((1 - ptr4_15slots).*sigma +
ptr4_15slots.*pds4_15slots.*Ts + ptr4_15slots.*(1 - pds4_15slots).*Tc);
s4_15slots = pds4_15slots.*ptr4_15slots.*e./t_slot4_15slots

% para n = 5
fun5_15slots = @solucao_ah_v15_qvariavel_n5est;
result5_15slots = fsolve(fun5_15slots,x0);
taud5_15slots = result5_15slots(1)
b0_05_15slots = result5_15slots(2)
n5_15slots = 5;

ptr5_15slots = 1 - ((1-taud5_15slots)^n5_15slots);
pds5_15slots = (n5_15slots*taud5_15slots*((1-taud5_15slots)^(n5_15slots-1)))/ptr5_15slots;
t_slot5_15slots = ((1 - ptr5_15slots).*sigma +
 ptr5_15slots.*pds5_15slots.*Ts + ptr5_15slots.*(1 - pds5_15slots).*Tc);
s5_15slots = pds5_15slots.*ptr5_15slots.*e./t_slot5_15slots;

% para n = 6
fun6_15slots = @solucao_ah_v15_qvariavel_n6;
result6_15slots = fsolve(fun6_15slots,x0);
taud6_15slots = result6_15slots(1)
b0_06_15slots = result6_15slots(2)
n6_15slots = 6;

ptr6_15slots = 1 - ((1-taud6_15slots)^n6_15slots);
pds6_15slots = (n6_15slots*taud6_15slots*((1-taud6_15slots)^(n6_15slots-1)))/ptr6_15slots;
t_slot6_15slots = ((1 - ptr6_15slots).*sigma +
 ptr6_15slots.*pds6_15slots.*Ts + ptr6_15slots.*(1 - pds6_15slots).*Tc);
s6_15slots = pds6_15slots.*ptr6_15slots.*e./t_slot6_15slots;

% para n = 7
```

```
fun7_15slots = @solucao_ah_v15_qvariavel_n7;
result7_15slots = fsolve(fun7_15slots,x0);
taud7_15slots = result7_15slots(1)
b0_07_15slots = result7_15slots(2)
n7_15slots = 7;

ptr7_15slots = 1 - ((1-taud7_15slots)^n7_15slots);
pds7_15slots = (n7_15slots*taud7_15slots*((1-taud7_15slots)^(n7_15slots-1)))/ptr7_15slots;
t_slot7_15slots =((1 - ptr7_15slots).*sigma +
 ptr7_15slots.*pds7_15slots.*Ts + ptr7_15slots.*(1 - pds7_15slots).*Tc);
s7_15slots = pds7_15slots.*ptr7_15slots.*e./t_slot7_15slots;

vazao15_5 = 5*s1_15slots;
vazao15_10 = 10*s1_15slots;
vazao15_20 = 5*s2_15slots + 10*s1_15slots;
vazao15_30 = 15*s2_15slots;
vazao15_40 = 10*s3_15slots + 5*s2_15slots;
vazao15_50 = 10*s3_15slots + 5*s4_15slots;
vazao15_60 = 15*s4_15slots;
vazao15_70 = 10*s5_15slots + 5*s4_15slots;
vazao15_80 = 10*s5_15slots + 5*s6_15slots;
vazao15_90 = 15*s6_15slots;
vazao15_100 = 10*s7_15slots + 5*s6_15slots;

%vazão qk
% para Nslot = 10 (10 slots dentro do raw)
% para n = 5
fun1 = @solucao_ah_v15_qk_n1;
result1 = fsolve(fun1,x0);
taud1 = result1(1);
b0_01 = result1(2);

% cálculo das probabilidades
ptr1 = 1 - ((1-taud1)^n1_15slots);
pds1 = (n1_15slots*taud1*((1-taud1)^(n1_15slots-1)))/ptr1;

% Cálculo das probabilidades de transmissão e de sucess
t_slot1 = ((1 - ptr1).*sigma + ptr1.*pds1.*Ts + ptr1.*(1 - pds1).*Tc);
s1 = (pds1.*ptr1.*e./t_slot1);

% para n = 10
fun2 = @solucao_ah_v15_qk_n2;
result2 = fsolve(fun2,x0)
taud2 = result2(1);
b0_02 =result2(2);

ptr2 = 1 - ((1-taud2)^n2_15slots);
pds2 = (n2_15slots*taud2*((1-taud2)^(n2_15slots-1)))/ptr2;
t_slot2 = ((1 - ptr2).*sigma + ptr2.*pds2.*Ts + ptr2.*(1 - pds2).*Tc);
s2 = (pds2.*ptr2.*e./t_slot2);

% para n = 20
fun3 = @solucao_ah_v15_qk_n5;
result3 = fsolve(fun3,x0);
taud3 = result3(1)
b0_03 = result3(2)


ptr3 = 1 - ((1-taud3)^n3_15slots);
pds3 = (n3_15slots*taud3*((1-taud3)^(n3_15slots-1)))/ptr3;
t_slot3 = ((1 - ptr3).*sigma + ptr3.*pds3.*Ts + ptr3.*(1 - pds3).*Tc);
s3 = pds3.*ptr3.*e./t_slot3;

% para n = 30
fun4 = @solucao_ah_v15_qk_n4;
result4 = fsolve(fun4,x0);
taud4 = result4(1);
b0_04 = result4(2);

ptr4 = 1 - ((1-taud4)^n4_15slots);
pds4 = (n4_15slots*taud4*((1-taud4)^(n4_15slots-1)))/ptr4;
t_slot4 = ((1 - ptr4).*sigma + ptr4.*pds4.*Ts + ptr4.*(1 - pds4).*Tc);
s4 = pds4.*ptr4.*e./t_slot4;

% para n = 40
fun5 = @solucao_ah_v15_qk_n5;
result5 = fsolve(fun5,x0);
taud5 = result5(1)
b0_05 = result5(2)

ptr5 = 1 - ((1-taud5)^n5_15slots);
```

```
pds5 = (n5_15slots*taud5*((1-taud5)^(n5_15slots-1)))/ptr5;
t_slot5 = ((1 - ptr5).*sigma + ptr5.*pds5.*Ts + ptr5.*(1 - pds5).*Tc);
s5 = pds5.*ptr5.*e./t_slot5;

% para n = 50
fun6 = @solucao_ah_v15_qk_n6;
result6 = fsolve(fun6,x0);
taud6 = result6(1)
b0_06 = result6(2)

ptr6 = 1 - ((1-taud6)^n6_15slots);
pds6 = (n6_15slots*taud6*((1-taud6)^(n6_15slots-1)))/ptr6;
t_slot6 = ((1 - ptr6).*sigma + ptr6.*pds6.*Ts + ptr6.*(1 - pds6).*Tc);
s6 = pds6.*ptr6.*e./t_slot6;

% para n = 60
fun7 = @solucao_ah_v15_qk_n7;
result7 = fsolve(fun7,x0);
taud7 = result7(1)
b0_07 = result7(2)

ptr7 = 1 - ((1-taud7)^n7_15slots);
pds7 = (n7_15slots*taud7*((1-taud7)^(n7_15slots-1)))/ptr7;
t_slot7 =((1 - ptr7).*sigma + ptr7.*pds7.*Ts + ptr7.*(1 - pds7).*Tc);
s7 = pds7.*ptr7.*e./t_slot7;

vazao15k_5 = 5*s1;
vazao15k_10 = 10*s1;
vazao15k_20 = 5*s2 + 10*s1;
vazao15k_30 = 15*s2;
vazao15k_40 = 10*s3 + 5*s2;
vazao15k_50 = 10*s3 + 5*s4;
vazao15k_60 = 15*s4;
vazao15k_70 = 10*s5 + 5*s4;
vazao15k_80 = 10*s5 + 5*s6;
vazao15k_90 = 15*s6;
vazao15k_100 = 10*s7 + 5*s6;


n = [n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11];
%vazao4 =((Tidle_15/beacon_interval)/1000000).*[5*s1_15slots,10*s2_15slots,15*s3_15slots,15*s4_15slots,
15*s5_15slots,15*s6_15slots,15*s7_15slots,15*s8_15slots,15*s9_15slots,15*s10_15slots,15*s11_15slots];
vazao4_qkn = ((Tidle_15/beacon_interval)/1000000).*[vazao15_5,vazao15_10,vazao15_20,vazao15_30,vazao15_40,
vazao15_50,vazao15_60,vazao15_70,vazao15_80,vazao15_90,vazao15_100];
vazao4_qk = ((Tidle_15/beacon_interval)/1000000).*[vazao15k_5,vazao15k_10,vazao15k_20,vazao15k_30,vazao15k_40,
vazao15k_50,vazao15k_60,vazao15k_70,vazao15k_80,vazao15k_90,vazao15k_100];

simulacaons3_15slots;

figure(1)
hold on
% plot(n,vazao1,'o-',n,vazao2,'*-.')
plot(n,vazao4_qkn,'bo-',n,vazao4_qk,'ks-',nrawsta,ns3_media15slots,'r*-')%,n,vazao1_razao_th,'bo-'
errorbar(nrawsta,ns3_media15slots,ns3_desvio15slots,'r')
%plot(n,vazao4)
grid on
%title('q dependendo de tau e p - 15 slots')
legend ('Nslot = 15 modelo qkn','Nslot = 15 modelo qk','Nslot = 15 NS3')
xlabel('Número de estações')
ylabel('Vazão Mbit/s')
```

As funções chamadas pelos programas principais são a solução da cadeia de Markov para o sistema não-linear resolvido pelo fsolve(). As funções são semelhantes, diferenciando apenas o número de estações contidas no slot RAW. Portanto, este é um exemplo e pode ser repetido para todo número de estações n contidas no slot RAW.

Função para probabilidade q dependendo do estágio de recuo e do número de estações contidas no slot RAW:

```
function F = solucao_ah_v2_qvariavel_n20(x)
taud = x(1);
b0_0 = x(2);

n = 10;
m = 6;

w0 = 16;
w1 = 2*w0;
w2 = 4*w0;
w3 = 8*w0;
w4 = 16*w0;
w5 = 32*w0;
w6 = 64*w0;
```

```
%prob_qvariavel;
%prob_qvariavel_2slots;
%prob_qvariavel_n10;

p = 1 - ((1-taud)^(n-1));
g = p;
%(1-p)^(n-1);


%vetores bi,j
b0_j = zeros(1,w0);
b1_j = zeros(1,w1);
b2_j = zeros(1,w2);
b3_j = zeros(1,w3);
b4_j = zeros(1,w4);
b5_j = zeros(1,w5);
b6_j = zeros(1,w6);


% estados bi,j
% N dos cálculos
n_linha0 = (1 - g*(1-q0));
n_linha1 = (1 - g*(1-q1));
n_linha2 = (1 - g*(1-q2));
n_linha3 = (1 - g*(1-q3));
n_linha4 = (1 - g*(1-q4));
n_linha5 = (1 - g*(1-q5));
n_linha6 = (1 - g*(1-q6));

b1_j(w1)= (p*(1-q1))/(w1*n_linha1).*b0_0;
b1_j_rev = wrev(b1_j);
for j = 1:1:w1

b1_j_rev(j+1) = (p*(1-q1))/(w1*n_linha1).*b0_0 + (((1-q1)*(1-g))/n_linha1).*b1_j_rev(j);

end
b1_0 =  b1_j_rev(w1);

b2_j(w2)= (p*(1-q2))/(w2*n_linha2).*b1_0;
b2_j_rev = wrev(b2_j);
for j = 1:1:w2

b2_j_rev(j+1) = (p*(1-q2))/(w2*n_linha2).*b1_0 + (((1-q2)*(1-g))/n_linha2).*b2_j_rev(j);

end
b2_0 = b2_j_rev(w2);

b3_j(w3)= (p*(1-q3))/(w3*n_linha3).*b2_0;
b3_j_rev = wrev(b3_j);
for j = 1:1:w3

b3_j_rev(j+1) = (p*(1-q3))/(w3*n_linha3).*b2_0 + (((1-q3)*(1-g))/n_linha3).*b3_j_rev(j);

end
b3_0 =  b3_j_rev(w3);

b4_j(w4)= (p*(1-q4))/(w4*n_linha4).*b3_0;
b4_j_rev = wrev(b4_j);
for j = 1:1:w4

b4_j_rev(j+1) = (p*(1-q4))/(w4*n_linha4).*b3_0 + (((1-q4)*(1-g))/n_linha4).*b4_j_rev(j);

end
b4_0 = b4_j_rev(w4);

b5_j(w5)= (p*(1-q5))/(w5*n_linha5).*b4_0;
b5_j_rev = wrev(b5_j);
for j = 1:1:w5

b5_j_rev(j+1) = (p*(1-q5))/(w5*n_linha5).*b4_0 + (((1-q5)*(1-g))/n_linha5).*b5_j_rev(j);

end
b5_0 = b5_j_rev(w5);

b6_j(w6)= (p*(1-q6))/(w6*n_linha6).*b5_0;
b6_j_rev = wrev(b6_j);
for j = 1:1:w6

b6_j_rev(j+1) = (p*(1-q6))/(w6*n_linha6).*b5_0 + (((1-q6)*(1-g))/n_linha6).*b6_j_rev(j);
```

```
end
b6_0 = b6_j_rev(w6);




%para o calculo de b0,0
soma_bi_0 = b0_0 + b1_0 + b2_0 + b3_0 + b4_0 + b5_0 + b6_0;
soma_bi_01 = (1-q0).*b0_0 + (1-q1).*b1_0 + (1-q2).*b2_0 + (1-q3).*b3_0 + (1-q4).*b4_0 + (1-q5).*b5_0 + (1-q6).*b6_0;
%soma_dados1 = b0_0 + sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
soma_dados1 = q1.*sum(b1_j_rev)+ q2.*sum(b2_j_rev)+ q3.*sum(b3_j_rev)+ q4.*sum(b4_j_rev)+ q5.*sum(b5_j_rev)+ q6.*sum(b6_j_rev);
%soma_dados = sum(b0_j_rev)+ sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
b_idle = soma_dados1;

m_linha = (1-p)*soma_bi_01 + b_idle;


b0_j_rev = wrev(b0_j);%inverte a ordem do vetor

b0_j_rev(1)= m_linha./(w0*n_linha0);% termo bo,w0-1,0
for j = 1:1:w0

b0_j_rev(j+1)= m_linha/(w0*n_linha0) + (((1-q0)*(1-g))/w0*n_linha0).*b0_j_rev(j);

end

%soma_dados1 = q0*sum(b0_j_rev)+ q1*sum(b1_j_rev)+ q2*sum(b2_j_rev)+ q3*sum(b3_j_rev)+
q4*sum(b4_j_rev)+ q5*sum(b5_j_rev) + q6*sum(b6_j_rev);
soma_dados = sum(b0_j_rev)+ sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
%s1 = ((1 - ((1-q0)^w0))/0.00001)*(1/w0);


%F(1)= soma_bi_0_k - taua;
F(1)= soma_bi_0 - taud;
%F(2)= 1 - ((1-taud).^(n-1))- p;
%F(2)=(((1/(1-g)*w0)*s1).*(soma_dados1)+((1-p)*soma_bi_0) + ((p*(1-q6))*b6_0)) - b0_0;
F(2)= (soma_dados + (b_idle))-1;
%F(4)= (1/w0)*s1.*(((1-pa)*(1-qa)*soma_bi_0_k) + ((1-p)*(1-q)*soma_bi_0) + q*b5_0) -b0_0;

end
```

Função para probabilidade q dependendo do estágio de recuo:

```
function F = solucao_ah_v2_qk_n20(x)
taud = x(1);
b0_0 = x(2);

n = 10;
m = 6;

w0 = 16;
w1 = 2*w0;
w2 = 4*w0;
w3 = 8*w0;
w4 = 16*w0;
w5 = 32*w0;
w6 = 64*w0;

prob_qvariavel;


p = 1 - ((1-taud)^(n-1));
g = p;
%(1-p)^(n-1);


%vetores bi,j
b0_j = zeros(1,w0);
b1_j = zeros(1,w1);
b2_j = zeros(1,w2);
b3_j = zeros(1,w3);
b4_j = zeros(1,w4);
b5_j = zeros(1,w5);
b6_j = zeros(1,w6);


% estados bi,j
```

```
% N dos cálculos
n_linha0 = (1 - g*(1-q0));
n_linha1 = (1 - g*(1-q1));
n_linha2 = (1 - g*(1-q2));
n_linha3 = (1 - g*(1-q3));
n_linha4 = (1 - g*(1-q4));
n_linha5 = (1 - g*(1-q5));
n_linha6 = (1 - g*(1-q6));

b1_j(w1)= (p*(1-q1))/(w1*n_linha1).*b0_0;
b1_j_rev = wrev(b1_j);
for j = 1:1:w1

b1_j_rev(j+1) = (p*(1-q1))/(w1*n_linha1).*b0_0 + (((1-q1)*(1-g))/n_linha1).*b1_j_rev(j);

end
b1_0 =  b1_j_rev(w1);

b2_j(w2)= (p*(1-q2))/(w2*n_linha2).*b1_0;
b2_j_rev = wrev(b2_j);
for j = 1:1:w2

b2_j_rev(j+1) = (p*(1-q2))/(w2*n_linha2).*b1_0 + (((1-q2)*(1-g))/n_linha2).*b2_j_rev(j);

end
b2_0 = b2_j_rev(w2);

b3_j(w3)= (p*(1-q3))/(w3*n_linha3).*b2_0;
b3_j_rev = wrev(b3_j);
for j = 1:1:w3

b3_j_rev(j+1) = (p*(1-q3))/(w3*n_linha3).*b2_0 + (((1-q3)*(1-g))/n_linha3).*b3_j_rev(j);

end
b3_0 =  b3_j_rev(w3);

b4_j(w4)= (p*(1-q4))/(w4*n_linha4).*b3_0;
b4_j_rev = wrev(b4_j);
for j = 1:1:w4

b4_j_rev(j+1) = (p*(1-q4))/(w4*n_linha4).*b3_0 + (((1-q4)*(1-g))/n_linha4).*b4_j_rev(j);

end
b4_0 = b4_j_rev(w4);

b5_j(w5)= (p*(1-q5))/(w5*n_linha5).*b4_0;
b5_j_rev = wrev(b5_j);
for j = 1:1:w5

b5_j_rev(j+1) = (p*(1-q5))/(w5*n_linha5).*b4_0 + (((1-q5)*(1-g))/n_linha5).*b5_j_rev(j);

end
b5_0 = b5_j_rev(w5);

b6_j(w6)= (p*(1-q6))/(w6*n_linha6).*b5_0;
b6_j_rev = wrev(b6_j);
for j = 1:1:w6

b6_j_rev(j+1) = (p*(1-q6))/(w6*n_linha6).*b5_0 + (((1-q6)*(1-g))/n_linha6).*b6_j_rev(j);

end
b6_0 = b6_j_rev(w6);




%para o calculo de b0,0
soma_bi_0 = b0_0 + b1_0 + b2_0 + b3_0 + b4_0 + b5_0 + b6_0;
soma_bi_01 = (1-q0).*b0_0 + (1-q1).*b1_0 + (1-q2).*b2_0 + (1-q3).*b3_0 + (1-q4).*b4_0 + (1-q5).*b5_0 + (1-q6).*b6_0;
%soma_dados1 = b0_0 + sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
soma_dados1 = q1.*sum(b1_j_rev)+ q2.*sum(b2_j_rev)+ q3.*sum(b3_j_rev)+ q4.*sum(b4_j_rev)+ q5.*sum(b5_j_rev)+ q6.*sum(b6_j_rev);
%soma_dados = sum(b0_j_rev)+ sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
b_idle = soma_dados1;

m_linha = (1-p)*soma_bi_01 + b_idle;


b0_j_rev = wrev(b0_j);%inverte a ordem do vetor

b0_j_rev(1)= m_linha./(w0*n_linha0);% termo bo,w0-1,0
```

```
for j = 1:1:w0

b0_j_rev(j+1)= m_linha/(w0*n_linha0) + (((1-q0)*(1-g))/w0*n_linha0).*b0_j_rev(j);

end

%soma_dados1 = q0*sum(b0_j_rev)+ q1*sum(b1_j_rev)+ q2*sum(b2_j_rev)+ q3*sum(b3_j_rev)+
 q4*sum(b4_j_rev)+ q5*sum(b5_j_rev) + q6*sum(b6_j_rev);
soma_dados = sum(b0_j_rev)+ sum(b1_j_rev)+ sum(b2_j_rev)+ sum(b3_j_rev)+ sum(b4_j_rev)+ sum(b5_j_rev) + sum(b6_j_rev);
%s1 = ((1 - ((1-q0)^w0))/0.00001)*(1/w0);


num_minislots_2slots;

%F(1)= soma_bi_0_k - taua;
F(1)= soma_bi_0 - taud;
%F(2)= 1 - ((1-taud).^(n-1))- p;
%F(2)=(((1/(1-g)*w0)*s1).*(soma_dados1)+((1-p)*soma_bi_0) + ((p*(1-q6))*b6_0)) - b0_0;
F(2)= (soma_dados + (b_idle))-1;
%F(4)= (1/w0)*s1.*(((1-pa)*(1-qa)*soma_bi_0_k) + ((1-p)*(1-q)*soma_bi_0) + q*b5_0) -b0_0;

end
```

Probabilidade q dependendo do estágio de recuo:

```
content...% probabilidade q váriavel a cada janela de contenção
% q aumenta a cada estágio de retransmissão

m = 6;

q0 = q*0;
q1 = q*1/(m+1);
q2 = q*2/(m+1);
q3 = q*3/(m+1);
q4 = q*4/(m+1);
q5 = q*5/(m+1);
q6 = q*6/(m+1);
```

Probabilidade q dependendo do estágio de recuo e do número de estações contidas no slot RAW:

```
% probabilidade q váriavel a cada janela de contenção
% q aumenta a cada estágio de retransmissão

m = 6;
constante_c;
n = 10;
fator_n10 = (n-1)/n;

q0 = fator_n10*c*0;
q1 = fator_n10*c*1/(m+1);
q2 = fator_n10*c*2/(m+1);
q3 = fator_n10*c*3/(m+1);
q4 = fator_n10*c*4/(m+1);
q5 = fator_n10*c*5/(m+1);
q6 = fator_n10*c*6/(m+1);
```

Constante$_c$:

```
c = 0.92;
```